Debreceni Egyetem
Informatika Kar

Adatbányászat Oracle - ben

Témavezető:
Dr. Ispány Márton
Egyetemi Docens

Készítette:
Szabó Zoltán
Programtervező Matematikus

Debrecen
2010

Tartalomjegyzék

Bevezetés

Ennek a diplomadolgozatnak a témája az adatbányászat. Kijelenthetjük, hogy eme összetett szó mögöttes tartalma sok érdekes témát foglal magába az informatika világában. Több különböző definíció közül tekintsünk meg kettőt.

„Az adatbányászat az új jelentéstartalommal bíró összefüggések, minták, trendek felfedezésének folyamata, amelyet nagyméretű tárhelyen tárolt adatok elemzésével valósít meg, felhasználva minta-felismerési technológiákat éppúgy, mint statisztikai és matematikai eszközöket.” /Gartner Group/

Ebből egyértelműen következtethetünk arra, hogy adatbányászat során nagy mennyiségű adatot dolgozunk fel komplex matematikai algoritmusok segítségével. Hogy ezt mi célból tesszük? Ennek megértésére ad támpontot a következő definíció:

„Újszerű, nem triviális, hasznos összefüggések keresése nagy adathalmazokban.

- Újszerű: nincs megelőző hipotézisünk, amit igazolni vagy cáfolni akarunk.
- Nem triviális: a magától értetődő összefüggéseket ki kell szűrni.
- Hasznos: üzleti igényt kell kielégíteni, nem öncélúan vizsgálódunk.
- Nagy adathalmaz: kis adathalmazon a statisztikai algoritmusok is vállalható időn belül lefutnak.”

A célunk az, hogy a meglevő nagymennyiségű adatból „információt” nyerjünk ki. Hogy mi a különbség egy adatbázis karbantartási tevékenysége és az adatbányászat között azt a következő szempontok alapján tudjuk összehasonlítani.

- Az adatok mennyisége.
- Feldolgozásra szánt idő.
- A figyelembe vett adatgyűjtési szempontok.
- Az eredményül kapott információk minősége.
- A felhasznált eszközkészlet komplexitása.

Az adatbányászatot nagyjából úgy kell elképzelni mintha egy ország lakosságát gép elé ültetnénk, és azt mondanánk nekik, hogy találják meg a rákkutatáshoz használható legjobb módszereket. Itt rögtön szembesülünk azzal, hogy a leírt probléma nagyon sokféleképpen megközelíthető, egyáltalán nem biztos, hogy a fellelt információ megbízható, jó minőségű és az eredmény rendszerezett formában elérhető. Valamint kétségeink támadhatnak a felől, hogy ezek az emberek záros határidőn belül értelmes és értelmezhető adatokat gyűjtenek össze.

Hogy egy vállalat, igazgató tanácsának – aki egy bizonyos problémakörrel kapcsolatos adatokból szeretne tájékozódni fontos döntések meghozatalakor – ne kelljen szembesülnie az információ kinyerésének nehézségeivel, létrejött az adatbányászat, mint az informatikának egy új területe.

Az adatbányászatnak három résztvevője van:

- Adatszolgáltató. Az az üzleti fél, akinek birtokában van a valószínűleg hosszabb ideje gyűjtött és tárolt információ, amely jellemezhet bizonyos például fogyasztói magatartást.
- Adatfeldolgozó. Az az informatikai fél, akinek birtokában van adatbányászat kivitelezéséhez szükséges számítástechnikai háttér, mind gépek mind szoftverek mind humán erőforrás tekintetében.
- Döntéshozó. Az az üzleti fél, aki hasznot akar húzni a kinyert információkból.

Egy adatbányászati projekt alapvetően több lépcsőből tevődik össze. Az adatbázisok tartalmazzák a rendszerezett adatokat. Ebből az adathalmazból kell nekünk kiválasztani azokat, melyeket fel akarunk használni. Azonban számos esetben az adatok minősége nem megfelelő, hiszen időnként hiányozhatnak lényeges attribútumok. Egyszóval ezek az anomáliák általánosak és gyakoriak, ezért tisztítani kell az adatokat, majd jöhet azok előfeldolgozása. Ahhoz hogy egy adatbányászati algoritmust hatékonyan tudjuk használni ahhoz az adatokat transzformálni kell valamint csökkenteni azok méretét. Tiszta, világos, jól determinált adatokon tudunk csak adatbányászni. Ez után az eredményképpen kapott információt érthető formában kell az üzleti fél rendelkezésére bocsátani, például grafikonok, táblázatok és riportok formájában.

Rendelkezésünkre áll egy informatikai eszköz, amely magába integrálja szinte a teljes informatikai oldal által nyújtott funkcionalitást, név szerint az Oracle Data Miner program.

Ez az eszköz (későbbiekben ODM)

- Használja az Oracle számtalan lehetőségét például adatimportálás bármilyen forrásból,
- Képes az importált adatok elő-feldolgozásra az Oracle Data Mining segítségével,
- Tartalmazza adatbányászathoz szükséges komplex matematikai algoritmusokat,
- Képes az eredmények exportálására.
- Programozható az ODM Java API-n keresztül.

Diplomamunkámban bemutatom, hogy milyen folyamatok vannak az adatbányászatban, milyen módszereket használnak és ezeket a módszereket hogyan segíti az ODM.

Összegyűjtöttem hogyan lehetne leírni az ODM főbb tulajdonságait. Az én szempontom alapvetően az, hogy mi az a problémakör melynek megoldására érdemes használni, és hogy hogyan érdemes ilyenkor használni ezt a terméket. Mik lehetnek azok az üzleti igények, melyekre válaszokat tudunk adni? És ezek megválaszolásához milyen típusú és mennyiségű adat szükséges.

Had említsek meg egy pár régi közhelyet, hogy világos legyen mire is gondolok:

„A cégek fuldokolnak az adatokban, de tudás-éhségben szenvednek."

„A számítógépről azt ígérték, hogy a bölcsesség forrása lesz, de csak az adatok áradatával találkozunk."

„Az elérhető információk mennyisége 20 hónaponként megduplázódik."

Alapvetően az a célunk, hogy megállapítsuk, hogyan lehet hatékony üzleti döntések meghozatalához értékes információt szolgáltatni. Az üzletfélnek anyagilag egyre olcsóbb megőrizni felhalmozott adatait, és éppen ezért egyre kifizetődőbb adatbányászati módszerekkel kinyerni azokból a hasznos információt. Folyamatosan bővül az adatbányászatban alkalmazott matematikai algoritmusok sokrétűsége, komplexitása, felkészültsége a különböző problémakörökre.

Láthatunk kialakulni egy világtrendet, nevezetesen hogy mindenki specializált, személyre-szabott szolgáltatást szeretne kapni, amit a cégek valahogy megpróbálnak kiszolgálni. Az információ éhség egyetemes probléma, melyre az adatbányászat megfelelő megoldást látszik nyújtani. A legnagyobb cégek is folytatnak adatbányászati fejlesztéseket, saját céljaiknak megfelelően például a Cisco, Google vagy Microsoft. Számomra az ODM megismerése megfelelő alapot jelentene, hogy elsajátítsam ezt a látványos fejlődésnek indult technológiát.

Véleményem szerint az adatbányászat legalább annyira összetett, mint maga az élet. Az adatbányászat eredményeit felhasználva következtetéseket levonva hozhatók döntések, melyek nagyban elősegíthetik akár egy cég gazdasági törekvéseit, akár egy orvostudományi labor DNS mintákkal foglalkozó kutatását.

Adatbányászat módszertana

A CRISP-DM [4] (*Cross Industry Standard Process For Data Mining*) módszer egy folyamat modell, amely egy általánosan használt szemléletmód, adatbányászatban megoldandó problémák esetén. 2002-ben, 2004-ben, 2007-ben, végzett közvélemény kutatások szerint ez a vezető metodológia melyet az adatbányászok használnak.

A jelenlegi folyamat diagram (1. ábra) áttekintést ad egy adatbányászati projekt életciklusáról. Tartalmazza a projekt megfelelő fázisait, a fázisok egyes feladatait, és a kapcsolatokat az egyes feladatok között. A diagram nem definiálja egzakt módon a kapcsolatokat és azok működését. Az, hogy mely fázisok közt definiálhatunk kapcsolatokat, az függ a projekt céljától vagy háttéranyagtól és az összefüggésektől mely iránt a felhasználó érdeklődik, de leginkább a rendelkezésre álló adatoktól függenek.

Egy adatbányászati projekt életciklusa 6 fázisból áll. Ezek között a fázisok között nincs pontos sorrend definiálva. Mindig szükséges lehet az előre hátra való lépés az egyes fázisok között. Az egyes fázisok eredményétől függ az, hogy melyik fázis, vagy egy fázis részfeladata kerül végrehajtásra. A nyilak csak a legfontosabb és leggyakrabban ismétlődő lépéseket jelzik a fázisok között.

A külső kör az adatbányászat természetes körforgását mutatja. Egy adatbányászati folyamat folytatódhat miután egy probléma megoldódott. A tapasztalatok megszerzése során újabb kérdések kerülhetnek megválaszolásra, gyakran a már meglévő problémakör pontosabb megismerése és üzleti kérdésekre fókuszálva. Az adatbányászat részfolyamatai profitálni fognak az előzőekben megszerzett tudásból. Az alábbiakban bemutatásra kerülnek az egyes fázisok:

1. Üzleti környezet megértése:

Ezen kezdeti fázis a projekt céljainak és követelményeinek megértését tűzi ki céljául. Ezután ennek alapján definiálni kell az adatbányászati problémát és egy előzetes tervet a célkitűzésekről.

2. Adatok megértése:

Ez a szakasz az adatok összegyűjtéséről és feldolgozásáról szól. Milyen adatok állnak rendelkezésre a vizsgálathoz. Meg kell vizsgálni, hogy az egyes adatok milyen értékeket

vehetnek illetve mennyire megbízhatóak. Esetleg a felderítés közben érdekes információk, kérdések esetleg hipotézisek kerülhetnek felszínre.

3. Adatok előfeldolgozása:

Ez a fázis magába foglalja, mindazokat a tevékenységeket (kiválasztás, átalakítás, tisztítás) mely során előáll a végső adathalmaz (ezek az adatok fognak alapjául szolgálni a modellező technikáknak) a kezdeti nyers adatból. Ezt a fázist egészen addig kell folytatni, míg az adathalmaz táblái, sorai, attribútumai nem lesznek megfelelőek az egyes modellező eszközök számára.

4. Modellkészítés:

Az adatbányászat legfontosabb művelete a modellkészítés. A modell lehet leíró (*descriptive*) vagy előrejelző (*predikitv*).

- A leíró modell segít megérteni az alapvető eljárásokat vagy viselkedéseket. például egy asszociációs modell leírja a fogyasztói magatartást.
- Az előrejelző modell egy egyenlet vagy szabályoknak egy olyan halmaza mely lehetővé teszi egy olyan érték előrejelzését mely nem szembeötlő vagy nem mérhető (függő változó, más néven kimenet). Mindehhez ismert értékeket (független változók, vagy bemenet) vesz alapul. Az egyenlet vagy a szabályok formája a bányászat közben gyűjtött adatok és azok tanulmányozásának eredménye. A jelenleg tanulmányozott adatbányászati folyamatból kigyűjtött adatok alapján áll elő az egyenlet vagy szabályok halmaza. Használhatunk tanulási vagy becslési technikát, hogy megbecsüljük az egyenlet vagy szabályok paramétereit. [5]

Ebben a fázisban történik a modellezési technikák kiválasztása, alkalmazása és azok paramétereinek kalibrálása a megfelelő értékre. Jellemzően, több technika létezik ugyanazon adatbányászati probléma megoldására. Ezek közül pár speciális követelményeket támaszt az adathalmazzal szemben. Pontosan e miatt, gyakran szükséges az előző lépésre való visszalépés.

5. Modellkiértékelés:

Ebben a fázisban a projekt már tartalmaz legalább egy modellt vagy modelleket mely(ek) feltételezik az adathalmaz megfelelő minőségét az adatok elemzése szempontjából. Mielőtt a

modell eredményinek bevezetése sorra kerülne, fontos a modell alapos kiértékelése, a modell létrehozása egyes lépéseinek felülvizsgálata, hogy biztosan jól megvalósítja az üzleti célokat.

6. Eredmények bevezetése:

Egy modell létrehozása még nem jelenti a projekt végét. Még akkor is, ha modell célja az adat pontos ismeretének növelése, a megszerzett tudást megfelelő formába kell hozni és prezentálni kell, hogy az az ügyfél számára használható legyen. Az elvárásoktól függően, a bevezetési fázis lehet igen egyszerű, például egy egyszerű riport vagy diagram elkészítése. Vagy esetleg lehet komplikáltabb, például egy ismétlődő adatbányászati folyamatot is meg lehet valósítani. Sok esetben ezt a megrendelő dönti el és nem az adatelemző, aki végrehajtja a bevezetési lépéseket. Mindazonáltal ha az adatelemző nem végezheti el ezt a lépést, akkor is fontos, hogy az ügyfél számára világos legyen milyen lépéseket kell végezni annak érdekében, hogy hasznosítani tudja a létrehozott modellt.

Oracle Data Mining áttekintés

Az Oracle Data Mining egy modul az Oracle relációs adatbázis kezelő rendszer *Enterprise Edition* verziójához. Ez az opció tartalmaz számos adatbányász és adatelemző algoritmust, például osztályozási (*classification*), becslési (*prediction*), regressziós (*regression*), csoportosítási (*clustering*), asszociációs (*association*), jellemzők kiválasztása (*feature selection*), anomáliák felfedezésére (*anomaly detection*), jellemzők kinyerése (*feature extraction*) vonatkozó problémák megoldására. Eszközöket nyújt létrehozásra, kezelésre és az adatbányászati modellek eredményeinek bevezetésére az adatbázis szerveren belül.

Az Oracle Data Mining hajtja végre a különféle adatbányászati algoritmusokat az Oracle relációs adatbázison belül, melyek implementációi közvetlen az Oracle adatbázis magjába vannak integrálva. Az algoritmusok működtetése natívan történik az adatbázis tábláiban tárolt adatokon. Ezzel elkerülhető az adatok exportálása és mozgatása más független adatbányász vagy elemző szervereken. A relációs adatbázis platform lehetővé teszi a biztonságos modell kezelést és SQL parancsok hatékony végrehajtását nagy adathalmazokon. A rendszer néhány általános művelet köré szerveződik, amelyhez egy általános és egységesített interfészt nyújt az adatbányászati funkciók eléréséhez. Ezek a műveletek magukba foglalják a következő funkciókat: létrehozás, alkalmazás, tesztelés és modellmódosítás. A modellek, mint adatbázisbeli objektumok vannak létrehozva és tárolva, azok menedzselése az adatbázisban

történik, hasonlóan, mint a táblák, nézetek, indexek és más adatbázisbeli objektumok kezelése.

A hagyományos analitikus alkalmazásoknál a már felépített és letesztelt modellt kell használnunk, hogy új adatokhoz jussunk vagy egy új relációs táblába mozgassuk az adatokat, az analitikus rendszer további feldolgozásra.

Az ODM egyszerűsíteti az adat kinyerési folyamatot az által, hogy SQL parancsokat használ az adatbázisban tárolt adatokon. Ily módon, mind a felhasználó mind az alkalmazás-fejlesztő képes dolgozni a létrehozott modellekkel és eredményekkel, tekintve hogy az Oracle relációs adatbázis kezelő rendszer képes folyamatos adatáramlásra és adatok manipulálása több lépésen keresztül, mind pedig párhuzamos és elkülönített adat hozzáférésre. [6]

Oracle Data Miner használata

Adatok megértése és transzformálása

A felhasznált adatokat az adatbányászati folyamathoz általában több helyről kell összegyűjteni, majd néhány átalakítást kell végre hajtani az adatokon, hogy adatbányászati műveletek alkalmazhatóak legyenek rajtuk. Az ODM nagy segítséget nyújt az adatok különböző forrásokból egy táblába való feltöltéséhez. Elvégezhető vele az adatok olyan formára való transzformálása, amely az egyes algoritmusok végrehajtásának előfeltétele. Ezeket az átalakítások (transzformációkat) a következő fejezetben mutatom be.

Az ODM a következő lehetőségeket nyújtja:

- Szöveges fájlból való adatok importálása egy adatbázis táblába.
- Az adatokon végrehajtott transzformációkat tárolhatjuk új nézetként vagy táblaként.
- Tábla létrehozására nézetből.
- Tábla másolás.
- Tábla vagy nézet eldobás.

Az importált adatokról megtekinthetőek összesítő statisztikai diagramok és különböző hisztogramok is. A rendelkezésre áll egy importálást végző varázsló mely különböző adatformátumokat képes rendkívül sokrétű feltételrendszernek megfelelően importálni. A következő lépésben áttekintjük, hogyan működik az importálást végző varázsló.

Importálás

Az import varázsló segítségével nagyon könnyű csv (*coma-separated*) fájlokat importálni. Használatakor az Oracle SQLLDR parancs hívódik meg, mely elérési útvonalát konfigurálni kell a beállításoknál. Használata igen egyszerű, először is ki kell keresni a feltöltendő file-t és meg kell adni a karakterkódolását. Majd ki kell választani a mező határolót és a mező elejét, végét jelző jelet. Feltöltés után az ODM automatikusan felismeri az oszlopneveket (ha azok meg vannak adva a file első sorában) és az attribútumok típusát, ez persze módosítható. Feltöltés után a NULL típusú mezők esetén be lehet állítani alapértelmezett értéket a '*Null If*' oszlopban. Így könnyedén ki lehet szűrni a hiányzó mezőket. Majd meg kell adni egy táblanevet ahova az adatok feltöltésre kerülnek vagy egy már létező táblának a nevét, ebben az esetben az adatok folytatólagosan lesznek feltöltve. Ezek után elkezdődhet az adatok feltöltése.

Táblák struktúrája és adatok összesítése

Az egyes táblák struktúrája és a benne lévő adatok megtekinthetőek, ha egyszerűen a tábla vagy nézet nevére klikkelünk. Itt láthatjuk sorrendben az elsődleges kulcsot, oszlop nevét, típusát, méretét, skáláját és hogy tartalmazhat-e az adott oszlop NULL elemeket.

A '*Data*' → '*Show Summary Single Record*' menüpont megnyitásával egy választott sémabeli táblának lehet megtekinteni az összesítő adatait. Ezen a képernyőn sok hasznos információt találunk. Minden egyes oszlop esetén látjuk az oszlop adatbányászatbeli és adatbázisbeli típusát, az elemek átlagát, maximum és minimum értékét, szórását és azt, hogy az adott oszlopbeli értékek közt van-e NULL elem. Minden egyes oszlopra klikkelve elérhető a '*Histogram*' menüpont, amellyel megtekinthető egy rövid statisztikai értékelés az adott oszlopbeli elemek eloszlásáról.

A '*Data*' → '*Generate SQL*' menüpont segítségével pedig nagyon egyszerűen tudjuk a létrehozott táblákat vagy nézeteket exportálni egy sql fájlba, amivel így az eredményünk nagyon könnyen kimenthető és hordozható.

Transzformációk

Számos transzformáció van beépítve az ODM-be. Minden egyes művelet alkalmazásakor az eredmény egy új nézetként jön létre alapesetben, de néhány transzformáció esetén lehetőség van új tábla létrehozására. A 2. ábrán megtekinthetők az alkalmazható transzformációk menüje.

- Aggregálás, (*Aggregate*):

Lehetővé teszi egy összegző függvény létrehozását, lényegében ez nem más, mint egy GROUP BY és HAVING záradék használatával egy új nézet létrehozása. És persze ennek a két záradéknak a paramétereit lehet specializálni egy dialóg segítségével, amelybe rengeteg beépített függvény található, amit a helyszűke miatt nem fejtenék ki bővebben.

- Új mező megadása (*Compute Field*):

Ezzel a transzformációval input adatból (bemenő adat mezői, nézet, tábla), lehet létrehozni új tartalommal és jelentéssel bíró oszlopot. Lényegében egy varázsló segítségével definiálhatóak új oszlopok egy már meglévő nézethez vagy táblához.

- Diszkretizálás (*Discretize*):

Diszkretizálás / kvantálás során szám típusú attribútumot kategória típusúvá alakítja a program. Az attribútum értékkészletét intervallumokra / csoportokra osztja és minden intervallumhoz egy kategóriát rendel. A diszkretizálás során nyilván információt veszítünk, viszont ezzel segíthető az adatbányász algoritmus.

- Egyedülálló rekord szűrése (*Filter Single Record*):

Ezzel a funkcióval egy nézet hozható létre úgy, hogy specializálni kell a kritériumot, amely szerint a tábla szelekciója történjen. Lényegében ez nem más, mint egy WHERE záradékkal megadott feltételrendszer definiálása. Szintén nagyon egyszerű, a már aggregálásnál megismert dialóg segítségével.

- Hiányzó Értékek Kitöltése (*Missing Values*):

Gyakran előfordulhat, hogy az adatok hiányosak (nem voltak mérhetőek, nem lettek megválaszolva, ismeretlenek vagy elvesztek). Adatbányászati módszerek eltérő módon kezelik a hiányzó értékeket. Jellemző módon, figyelmen kívül hagyják a hiányzó értékeket, elhagyják a hiányzó értékeket tartalmazó rekordokat vagy helyettesítik a hiányzó értékeket, vagy következtetnek azokra a meglévő értékekből.

- Normalizálás (*Normalize*):

Normalizáláson azt értjük, hogy az attribútum elemeit egy másik intervallum elemeivel helyettesítjük úgy, hogy a helyettesített értékek eloszlása megegyezzen az eredeti értékek eloszlásával. Tegyük fel, hogy az A attribútum $a_1, a_2, \ldots, a_k$ értékeket veszi fel. Az $a_j, j = 1, \ldots, k$ érték normalizáltját $a_j^{'}$-vel jelöljük. Normalizálásra három módszer áll rendelkezésre az ODM-ben.

- $(x - MIN(x))/(MAX(x) - MIN(x)) * (new\max - new\min) + new\min$

  A megadott minimum és maximum értékek közé való normalizálás.
- $(x - AVG(x))/SQRT(VARIANCE(x))$

  A 0-tól és az 1-től való eltérés középértékébe való normalizálás. ($VARIANCE(x)$-szórásnégyzet)
- $(x / MAX(ABS(MIN(x)), ABS(MAX(x))))$

  -1 és 1 intervallumba való normalizálás.

- Számszerűsítés (*Numeric*):

Ezzel a transzformációval az adatok eloszlását lehet megváltoztatni egy előre definiált séma segítségével. Ha egy attribútum adataira akarjuk alkalmazni úgy, hogy nem veszítjük el az eredeti adatokat akkor érdemes előtte az új mező transzformációt alkalmazni. Az újonnan létrehozott attribútumra pedig alkalmazható ez az átalakítás. Numerikus típus esetén érdemes diszkretizálni az adatokat a gyorsabb végrehajtás érdekében.

Stabilizálja az ingadozó szórásnégyzetet és megközelítő megoldást ad összetett eloszlások esetén, különösen, amikor a múltbeli eloszlás nincs hatással a jövőbelire. A következő eloszlások használhatóak ennél a transzformációnál: $EXP(x)$, $LN(x+a)$, $LN((x-a)/(b-x))$, $LOG(10, x+a)$, $SQRT(x)$, $1/EXP(AVG(x)-x)$

- Kilógó adatok kezelése (*Outlier Treatment*):

Kilógónak nevezünk egy adatpontot, amely távol esik az adathalmaztól. Tipikusan különböző adatokat kell érteni, melyek a középértéktől távol esnek. Ez a transzformáció a kilógó adatok kezelésére előállít egy – a kiválasztott adatbányász algoritmushoz – ajánlott eljárást. Egyes algoritmusok megkövetelik ennek a transzformációnak a használatát. Néhány adatbányász algoritmus érzékeny az ilyen fajtájú adatokra. Van lehetőség egyéni eljárásokat is definiálni, például az adatoknak százalékban megadott, felső és alsó része vagy egy megadott értéknél nagyobb illetve kisebb értékek lecserélhetők NULL értékekre vagy a határértékekre.

- Újrakódolás (*Recode*):

Az újrakódolás lehetőséget ad arra, hogy egy meghatározott attribútum értéket helyettesítsünk egy új értékkel. Működése során beolvassa egy attribútum értéktartományát, és erre összeállítható egy egyedi feltételekből álló lista, amelyet a transzformáció befejezése után a rendszer kiértékel. Ennek a transzformációnak a használata erőforrás-problémákat eredményezhet amennyiben az attribútum numerikus típusú és folytonos. Ez elkerülhető, ha diszkretizálással csoportokra bontjuk az adathalmazt.

- Mintavételezés (*Sample*):

Ez a transzformáció egy táblát vagy nézetet hoz létre véletlenszerűen kiválasztott adatokból. A mintavételezés történhet táblából vagy nézetből. A minta mérete a rendelkezésre álló

rekordok számával vagy százalékának megadásával határozható meg. Megadható egy véletlen szám. amelyet a véletlenszerűsítéshez használ. Egy mintavételezés általában kisebb, mint az eredeti adathalmaz, amelyet forrásnak használtunk. Teljesítménynövekedést érhetünk el, ha mintavételezést használunk. Ha egy minta megfelelően reprezentálja a teljes adathalmazt, akkor egy a mintán előállított eredmény kompatibilis lesz a teljes halmazon előállított eredménnyel.

- Rétegzett mintavételezés (*Stratified Sample*):

Ez a transzformáció abban különbözik a normál mintavételezéstől, hogy kiválasztható egy attribútum, majd az attribútum különböző értékeire megadható hogy hány vagy milyen arányban forduljanak elő a mintában.

- Kettéosztás (Split):

Ez a transzformáció egyértelmű. Egyszerűen ki kell választani azt a táblát vagy nézetet melyet vizsgálunk. Meg kell adni, hogy milyen arányban legyen kettéosztva az adathalmaz, majd befejezzük a transzformációt.

- Rétegzett kettéosztás (*Stratified Split*):

A kettéosztás ezen fajtája szintén annyiban különbözik a normáltól, hogy a varázsló felajánlja egy attribútum kiválasztását, mely szerinti ossza ketté az adatokat.

- Szórás szűrés (*Variation Filter*):

Ezzel a transzformációval definiálható egy szűrő, mellyel kiszűrhetők olyan esetek melyek nem tűnnek ígéretesnek. Ez a szűrő általában bioinformatikában és kemoinformatikában használható. Ezekben a speciális esetekben néha szükségessé válik olyan esetek felkutatása – génkutatásban, vagy vegyi reakciók elemzése esetén – ahol az értékek meghaladnak egy bizonyos arányt vagy különbséget. A rekordokat a $\max/\min$ vagy $\max-\min$ szűrők segítségével transzformálhatjuk.

- Szöveges transzformáció (*Text*):

Az ODM osztályozó, regressziós, anomáliák felfedezése, tulajdonságok kinyerése, és K-középpontú csoportosítási modellek készítése esetén engedi azt, hogy a bemenő adathalmaz szöveges oszlopot tartalmazzon.

Mielőtt egy táblát, amely szöveges oszlopot tartalmaz, adatbányász művelethez használunk, kötelező transzformálni azokat az oszlopokat, melyeket fel akarunk használni és szöveg típusúak. A mód, ahogy a tábla szöveges oszlopa elő lesz készítve, függ attól, hogy az adatok hogyan lesznek felhasználva. Ugyanilyen módon kell transzformálni ezt az oszlopot, mely a modell készítése (*Build*), a modell tesztelése (*Test*) és a modell alkalmazása (*Apply*) folyamatain keresztül használunk. Ez a transzformáció a következő menüpontból érhető el: Data → Transform → Text.

Két beállítással futtatható ez a transzformáció:

- Magyarázó és leképező táblák létrehozása (*Create Explain and Mapping Tables*): Ebben az esetben egy szöveg típusú oszlopot transzformálunk. Ez az alapértelmezett beállítás. Ez a varázsló egy szöveges oszlopot dolgoz fel a folyamat során és összegyűjti a tulajdonságokat a kiválasztott szöveges oszlopból és indexeli a tartalmi összefüggéseket.

  A transzformáció eredménye két tábla:

  - ❖ Magyarázó tábla (*Explain table*): Tranzakciós formátumú tábla mely minden egyes attribútum azonosítót összekapcsol egy értékkel
  - ❖ Leképező tábla (*Mapping table*): Ebben a táblában vannak összerendelve az attribútum azonosítók és a szöveges tulajdonságok

    Mivel az Oracle Data Mining Java interfésze megköveteli, hogy a szöveges adatok táblákba legyenek rendezve ezért ennek a varázslónak az eredménye egy táblában tárolódik.
- Szöveg transzformálása beágyazott oszlopként (*Transform text into nested column*): Ebben az esetben egy vagy több szöveg típusú oszlop transzformálható beágyazott táblákba. Így ezek később felhasználhatóak mind PL/SQL, mind Java oldalon.

  Ez a varázsló a kiválasztott szöveg típusú oszlopot beágyazott tábla típusúvá transzformálja, úgy hogy az tartalmazza a numerikus típusú (*DM_Nested_Numericals*) vagy a kategória típusú (*DM_Nested_Categoricals*) elemeket. Az átalakított oszlop használható PL/SQL programból.

  Megjegyzés: Beágyazott tábla nem használható adatbányászati algoritmus bemeneteként. Maximálisan egy szöveg típusú oszlopot tartalmazhat egy tábla melyet egy eljárás használ.

ODM erőteljes és hasznos eszközrendszerrel rendelkezik analízisre, aggregálásra, adatok előkészítésre és transzformálására. Lehetőség van a diszkretizálás testre szabására, hogy illeszkedjünk az üzleti igényekhez. Új adatok létrehozására és az SQL-ben lévő nagyszerű

lehetőségek kihasználására. ODM mint eszköz automatizálja a fontos funkciókat, melyek szükségesek egyes adatbányász algoritmusok alkalmazásához. A transzformációs varázsló kellő rugalmassággal kezeli az adathalmazok módosítását, statisztikai eljárásokkal emeli az adatok minőségét és javítja az osztályozási modellek becslési pontosságát. Minden egyes művelet esetén lehetőség van a transzformáció eredményének megtekintésére, így ha azok nem teljesítik az elvárásokat, akkor még módosítható a transzformáció.

Adatbányászati tevékenységek ismertetése

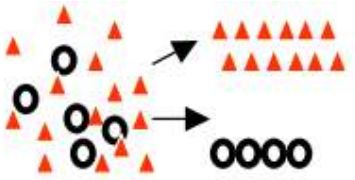
Amikor az adatbányászati problémát megfogalmaztuk, és annak megoldására rendelkezésünkre áll a kellő minőségű adathalmaz, akkor már csak két fázis van hátra: Egy megfelelő modell építése és annak kiértékelése.

Az ODM tartalmaz egy varázslót mely sokban segíti a felhasználót eme két fázis kivitelezésében. Sőt mi több ez a varázsló zökkenőmentesen végigvezeti a felhasználót modell készítése (*Build*), modell tesztelése (*Test*) és a modell alkalmazása (*Apply*) folyamatain.

Választhatjuk azt, hogy a szoftver saját belső optimalizálója állítsa be az egyes algoritmusok paramétereit, így nekünk csak az adatokat kell kiválasztani és az algoritmust megadni. Ha viszont valaki, tisztában van a paraméterek módosításának hatásával, az választhatja azt, hogy egyénileg állítja be a paramétereket és kézzel módosítja a műveleteket.

Ez a fejezet mutatja be, hogy néz ki és milyen lépésekből áll ez a varázsló (*Activity Guide Wizard*). Az egyes algoritmusokban felajánlott választási lehetőségek és a paraméterek jelentése illetve mögöttes okai megvitatásra kerülnek.

Osztályozási algoritmusok (*Classification*)



Az osztályozás lényege, hogy az általunk definiált kimeneteknek megfelelően részhalmazokra bontja a tárolt adatokat. Az osztályozási technikák lehetővé teszik az adatokban rejlő összefüggések felismerését, ezért az osztályozás hatékony művelet lehet például a direkt marketing területén. Alkalmazása során olyan információ birtokába juthatunk, amely nagymértékben megkönnyíti a jövőre vonatkozó döntéshozatalunkat, vagyis egy pontos osztályozási modell segítségével az általunk vizsgált attribútumok (a modell cél-célattribútumai) értékei megjósolhatóvá válnak. Például, ha ismerjük azt a tényt, hogy a 90 ezer Ft-nál magasabb jövedelmű, nős férfiak 60%-a (ami az

összes érintett személy 3%-át teszi ki) pozitívan reagált egy előző reklámkampányra, jó eséllyel feltételezhetjük, hogy egy hasonló paraméterekkel rendelkező új ügyfél számára is megnyerő lesz a termékünk. Más szóval ismeretlen, előre nem megfigyelhető változók, attribútumok értékének előrejelzése más ismert, megfigyelhető változók, attribútumok ismeretében.

Adatbányászati modelleket alapul véve két fajta tanulás létezik, felügyelt és nem felügyelt. Felügyelt tanulás esetén tipikusan értékek előrejelzéséről van szó. Az osztályozási problémák megoldása felügyelt tanulással történik. Ilyen esetben rendelkezésünkre áll számos eset és azt szeretnénk megjósolni, hogy az egyes esetek melyik osztályba tartoznak. Minden egyes eset több attribútumból áll, az egyes attribútumok több lehetséges értéket vehetnek fel. Egy attribútum tartalmaz több előrejelző attribútumot (független változók) és egy cél attribútumot (függő változó). Minden egyes cél attribútum egy lehetséges osztályozást ad meg, amelyet az egyes esetek előrejelző attribútum-értékei alapján lehet megjósolni. Különböző osztályozási algoritmusok különböző technikákat használnak arra, hogy kapcsolatokat találjanak az előrejelző attribútum értékei és a cél-attribútum értékei közt a felhasznált adatokban, hogy előállítsák a modellt. Ezek a kapcsolatok vannak összefoglalva a modellben. Ezt a modellt lehet alkalmazni új esetekre melyek, nem tartalmaznak cél értékeket, hogy megjósolja azokat. Az osztályozási modell új adatokra való alkalmazását úgynevezett modell alkalmazásnak (*applying the model*) vagy adatok pontozásának (*scoring data*) nevezzük.

Egy osztályozási probléma esetén fontos lehet meghatározni a költségeket, hogy elkerüljük a hibás döntéseket. Ennek során hasznos lehet, ha a különböző besorolások költségei jelentősen eltérőek. A költségeket egy költség mátrixban lehet meghatározni. A költség mátrix sorai az aktuális értékeknek, az oszlopok pedig a várható értéknek felelnek meg. Minden egyes aktuális és becsült értékpáros esetén, a mátrix tartalmaz egy bejegyzést az osztályozás során előálló párok számáról.

Amikor egy osztályozó modellt használunk, szükséges lehet egyenlő mértékben definiálni pozitív és negatív eseteket a célpont algoritmus számára. Ez vagy úgy történhet meg, hogy a megadott célérték egy ritkán előforduló érték. Vagy azért mert a rendelkezésre álló adatok nem pontosan tükrözik a valós sokaságot, ez az az eset, amikor a mintavételezés szimmetrikus. A prior-ok által megadható hogy hogyan állíthatunk elő adatokat, melyek tulajdonságaikban megegyeznek az összességében meglevőkkel.

Az osztályozó algoritmusok akkor működnek a leghatékonyabban, ha megvan a megfelelő számosságú példa minden egyes célértékhez a felépített táblában. Amikor csak néhány lehetséges célérték létezik, nem fogunk pontos eredményt kapni.

Hogy megoldjuk ezt a problémát egy tanuló táblát kell létrehozni pozitív és negatív cél értékekkel melyek többé-kevésbé egyenlő számosságban vannak jelen, majd ezek után a prior-ok által szolgáltatott információk adják meg, a modellt.

Az osztályozási modell készítése varázsló a következő algoritmusokat nyújtja a felhasználók számára:

1. Naiv Bayes algoritmus

A Naiv Bayes algoritmus osztályozási problémák megoldására használható. Ez az eljárás a Bayes-tételt használja fel, mely a valószínűségszámításban egy feltételes valószínűség, és a fordítottja között állít fel egy összefüggést.

A Bayes-tétel segítségével meghatározható az optimális klasszifikációs szabály. Jelöljük A_i-vel azt, amikor a klasszifikálandó eset az *i*-edik osztályba tartozik. Az elemek megfigyelhető tulajdonságait a B vektor írja le. Ekkor egy ismeretlen, A tulajdonságú példányt abba az osztályba érdemes sorolni, amelyikre $P(A_i \mid B)$ maximális. A Bayes-tétel alapján

$$P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{P(B)}$$

$P(A_i)$-t az A_i eseményt a *priori*, $P(B \mid A_i)$-t a *posteriori* valószínűségeknek is nevezik; a szokásos értelmezésben A_i a hipotézis, B egy megfigyelhető esemény, és a tétel azt adja meg, hogyan erősíti vagy gyengíti az esemény megfigyelése a hipotézis helyességébe vetett hitünket.

Mivel $P(B)$ minden *i*-re konstans, ezért elegendő $P(B \mid A_i)P(A_i)$-t maximalizálni. $P(A_i)$ adott, vagy pedig a mintából a relatív gyakoriságokkal egyszerűen becsülhető. Így a $P(B \mid A_i)$ marad, amit meg kell határozni. Amennyiben *k* darab bináris magyarázó attribútum van megadva, az A pedig l értéket vehet fel, akkor $l(2^k - 1)$ darab $P(B \mid A_i)$ értéket kell megbecsülni.

A Naiv Bayes algoritmus olyan feltételezéssel él, amelynek segítségével a $l(2^k - 1)$ darab megbecsülendő paraméter száma $l * k$-ra csökkenthető. A feltételezés szerint egy osztályon

belül az attribútumok feltételesen függetlenek egymástól. Ekkor a $P(B \mid A)$ valószínűség kifejezhető a $P(B_i \mid A)$ valószínűségek szorzataként, hiszen:

$$P(B_1, B_2 \mid A_i) = P(B_1 \mid B_2, A_i)P(B_2 \mid A_i) = P(B_1 \mid A_i)P(B_2 \mid A_i).$$

És így k darab magyarázó változó esetén a következőt kapjuk:

$$P(B_1, B_2, \ldots, B_k) = P(x_1, x_2, \ldots, x_k \mid A_i) = \prod_{j=1}^{k} P(B_j = x_j \mid A_i)$$

Amennyiben B_j kategória típusú, akkor $P(B_j = x_j \mid A_i)$ valószínűséget a relatív gyakorisággal közelítjük, vagyis meghatározzuk a relatív arányát a B_j attribútumában x_j értékét felvevő elemeknek az A_j osztályú elemek között.

Amennyiben B_j numerikus típusú és tudjuk a $P(B_j \mid A_i)$ eloszlásának típusát, akkor a keresett valószínűséghez szükséges eloszlás-paramétereket statisztikai módszerrel becsüljük.

A Naiv Bayes algoritmus megvizsgálja az adatokat, és azok alapján kiszámítja a cél-attribútum értékeinek feltételes valószínűségét, megfigyelve az attribútum értékek gyakoriságát és azok kombinációjának gyakoriságát.

A Naiv Bayes osztályozó hátránya, hogy feltételes függetlenséget és egyenlőséget feltételez. Sokat javíthatunk a Naiv Bayes osztályozók pontosságán, ha előfeldolgozás során meghatározzuk a fontos attribútumokat, amelyekről úgy gondoljuk, hogy nem függetlenek az osztályattribútumtól. [9]

Naiv Bayes algoritmus ODM-beli használata:

Tegyük fel, hogy A jelöli azt, hogy „a vevő emelte a költségvetését” B pedig azt, hogy „a vevő házas”, és mi annak valószínűségét szeretnénk meghatározni, hogy egy házasságban élő vevőnek a kiadásai emelkedni fognak. Ennek a kérdésnek a megválaszolásához számos tényezőt kell figyelembe venni. Például a vevő házas, 35 és 25 év közötti, 3 vagy 4 gyereke van, és tavaly Y terméket megvásárolta, és így tovább. Az algoritmusnak jelentős mennyiségű esetet kell figyelembe vennie ahol A és B események egyszerre fordul elő. Ezek lesznek a gyakori elempárok, melyek százalékos arányban vannak kifejezve. Ha ez az arány nem megfelelő mértékű, akkor ronthatják a modell hatásfokát. Pontosan ezért a sebesség és pontosság növelése érdekében egy bizonyos határérték alatt ezeket az előfordulásokat az algoritmus figyelmen kívül hagyja.

Az a megoldandó probléma, hogy megkülönböztessük a vásárlóinkat abból a szempontból, hogy mennyire megbízhatóak és ragaszkodóak. Abból a célból, hogy egy kérdőívet töltessünk ki velük és a visszaérkező válaszok mennyisége maximalizálva legyen.
Az Oracle hivatalos ODM kézikönyve alapján fogom bemutatni a Naiv Bayes algoritmus működését. (Online elérhetőség hivatkozása: Irodalomjegyzék [7], pontosabban annak 5. fejezete és A, B függeléke) Az adattábla struktúrája, melyen a Naiv Bayes algoritmust teszteltem, a 3. ábrán tekinthető meg.

Első lépés a modell készítése (*Activity → Build*). A paraméterek megadásában egy varázsló fog segítséget nyújtani. Miután kiválasztottuk az osztályozást majd a Naiv Bayes algoritmust (4. ábra) lépjünk is a következő oldalra.

Második lépésben (5. ábra) ki kell választani azt a sémát és táblát, amelynek az adatait fel akarjuk használni. A kiválasztott tábla egyedi azonosítója (*Unique Identifier*) megadható, ha nincs elsődleges kulcsa vagy összetett kulccsal rendelkezik, akkor a második opciót kell választani (*Compound, or None*). Meg lehet határozni, hogy a tábla mely oszlopait használjuk fel az analízisre. Van lehetőség további attribútumok hozzáadásra, amely a jelölőnégyzet (*Join additional data with case table*) kijelölésével hozzáférhetővé válik. Válasszuk ki a „*MINING_DATA_BUILD_V*” táblát.

Harmadik lépésben (6. ábra) meg kell határozni a cél-attribútumot és azokat az attribútumokat, melyeket fel akarunk használni a modellezés alatt. Megjegyzés: előfordulhat olyan eset mikor a mintavételezés során egy attribútumból csak egyféle értékkel rendelkező rekordokat sikerült kiválasztani, ilyenkor az ODM figyelmeztet, hogy ez az érték a mintában konstans értékkel rendelkezik. Ez az attribútum ronthat a modell hatásfokán. A jelenlegi cél az, hogy megkülönböztessük a vásárlóinkat abból a szempontból, hogy mennyire megbízhatóak és ragaszkodóak. Ezt az információt az ’*AFFINITY_CARD*’ attribútum tartalmazza (1 = lojális vásárló, 0 = kevésbé lojális vásárló). A cél-attribútumnak válasszuk az ’*AFFINITY_CARD*’ attribútumot.

Negyedik lépésben (7. ábra) azt a cél-attribútum értéket kell kiválasztani a legördülő menüből, amelyhez kapcsolódó összefüggéseket szeretnénk megvizsgálni. Ennek az értéknek a megadása a modell tesztelésénél bír jelentőséggel. A modell elkészülte után, lehetőségünk

van a célérték módosítására és újratesztelésre. Mivel minket most azok a vásárlók érdekelnek, akik megbízhatóak ezért válasszuk ki az 1-es értéket.

Ötödik lépésben megadható a modell neve és megjegyzése. Ebben a lépésben lehet módosítani a modellkészítés paramétereit, mely haladók számára ajánlott. Ha megtekintjük a felugró dialógot (*Advanced Settings…*), öt fület láthatunk, melyeken az egyes transzformációkat illetve a modell készítése és tesztelése műveleteket lehet módosítani. Nézzük végig a Naiv Bayes algoritmus haladó beállításait.

- Az első fülön (*Sample*) a mintavételezéssel kapcsolatos beállításokat találjuk. Az algoritmusok alapesetben nem használnak mintavételezést, Az Oracle Data Mining „bármekkora" adathalmazt képes skálázni. Viszont ajánlott a mintavételezés funkciót bekapcsolni, ha kellően nagy adathalmaz esetén azt szeretnénk, hogy az algoritmus emberi időn belül lefusson. Így a hardveres erőforrás felhasználása is kisebb lesz. Ha a mintavételezést bekapcsoljuk, akkor kiválasztható a mintavételezés mérete és annak módszere. A véletlenszerű (Random) mintavételező úgy választ, hogy a mintában lévő cél-attribútum értékeinek eloszlása hozzávetőlegesen ugyan olyan legyen, mint az eredeti adathalmazban. Rétegzett mintavételezés esetén, körülbelül azonos számú eset kerül kiválasztásra minden cél érték esetében. Ez a fajta mintavételezés akkor lehet hasznos, ha például a probléma rendellenes esetek felfedezése vagy ritka betegségek kimutatása.

- Második fülön (*Discretize*) a diszkretizálási stratégiákat lehet kiválasztani numerikus és kategória típusú attribútumok esetén. A numerikus típusú attribútumokat értéktartományokba, a kategória típus esetén pedig minden egyes értéket egy csoportba (*bin*) sorol. Azokkal az értékekkel, amelyeket nem tudott diszkretizálálni, azokat egy '*Other*' címkéjű csoportba helyez. Minden egyes csoportot egy egész számmal jelöl. Ez a lépés a Naiv Bayes algoritmus számára azért lényeges mert, egyfajta számlálási technikát használ, hogy kiszámolja a valószínűségeket. Így sokkal könnyebb egész típusú számokkal dolgozni, mint valós vagy karakter típusúakkal.

  Numerikus típusok esetén a következő stratégiák választhatók:

  ❖ Az osztóérték (*Quantile*) csoportosítás, hozzávetőlegesen annyi csoportot hoz létre ahány fajta érték előfordul az attribútum érték-tartományában függetlenül attól, hogy hány rekord veszi fel ugyanazt az értéket.

- ❖ Egyenlő szélességű (*Equi-width*) csoportosítás, azonos szélességű csoportokat hoz létre, függetlenül az előforduló az attribútum érték számosságától. Valójában, ez a módszer létrehozhat üres csoportokat is.

- Harmadik fülön (*Split*) a meglévő adathalmazt szétoszthatjuk két részre. Az első részét az algoritmus, mint megtörtént eseményeket, fogja elemezni és abból építi fel a modellt. a második részt pedig, mint jövőbeli események, tesztelésre fogja felhasználni. Alapértelmezett módon az algoritmus az adatok 60%-át használja a modell építésre, és 40%-át pedig a modell tesztelésére.

- Negyedik fülön (*Build*) található két menüpont közül az általánoson (*General*) a maximum átlagos pontosság és a maximum általános pontosság opciók közül lehet választani. A második menüpont az algoritmus beállításait (*Algorithm Settings*) lehet pontosítani:
  - ❖ Egyedülálló küszöbérték (*Singleton threshold*): Egy elem gyakoriságának számlálására szolgáló százalékos érték. Tegyük föl, hogy a tranzakciók számossága k és a profilok száma P és t az egyedülálló küszöbérték mely P százalékos arányában van kifejezve. Akkor nevezünk egy elemet gyakori elemnek, ha teljesül rá, hogy $k \geq t * \mathrm{P}$.
  - ❖ Páronkénti küszöbérték (*Pairwise threshold*): Szintén a gyakoriság számlálására szolgáló százalékos érték, elempárok esetén. Gyakori egy elempár, ha a profilban elég gyakran fordulnak elő együtt. Tegyük fel, hogy a két különböző elem k profilban fordul elő egyszerre, az összes profilból. Legyen P, a gyakori elemeket is tartalmazó, profilok halmaza, és t a küszöbérték mely P százalékos arányában van kifejezve. Akkor nevezünk egy párt gyakori elempárnak, ha teljesül rá, hogy $k > t * \mathrm{P}$.

- Ötödik fülön (*Test Metrics*) beállítható, hogy a tesztelés eredményeként milyen grafikonokat szeretnénk látni. Az egyes beállítási lehetőségek bemutatásra kerülnek, mikor az eredményeket fogjuk feldolgozni. Itt mind a '*Lift Result*' mind pedig a ' *ROC Result*'-hoz tartózó jelölőnégyzetet kijelöljük.

Most hogy készen vagyunk a Naiv Bayes algoritmus beállításaival, létrehozhatjuk a modellünket. A következő lépésben (8. ábra) láthatjuk, hogy az ODM lépésről lépésre halad az egyes transzformációkon és műveleteken. Amíg az ODM ezeken dolgozik láthatjuk, hogy melyik transzformációt vagy műveletet hajtja éppen végre az ODM. Egy állapot sáv jelzi, hogy hány százalékban készült el a modell. Ha egy lépés hibával ért véget, akkor egy hiba

üzenet megjelenik a képernyőn, ilyenkor át lehet konfigurálni az egyes transzformációkat vagy műveleteket, majd újra lehet futtatni a modellt.

Naiv Bayes eredményeinek megtekintése

A futás végeztével a modell és a tesztelés eredménye megtekinthető, a '*Build*' szekcióban az eredmények (*Result*) menüpontra klikkelve. A felugró dialóg a következő füleket tartalmazza:

- Feladatok (*Task*): Ezen a fülön az általános információkat tekinthetjük meg, például modell neve, indítás és befejezés időpontja, bemenő és kimenő adatok.

- Modell építésének beállításai (*Build Settings*)
  Itt láthatjuk a modell típusát (jelen esetben osztályozási feladat). Az algoritmust, melyet a modell építésére használtunk, az algoritmus paramétereit, a modell cél-attribútumát és a többi attribútumot, melyeket a modell építésére használtunk.

- Gyakori elempárok előfordulásának valószínűsége (*Pair Probabilities*)
  Ez a fül tartalmazza (9. ábra) a gyakori elempárok listáját, a valószínűségi érték egy a modell készítése során használt attribútum és a cél-attribútum kombinációja. Láthatjuk a cél-attribútum értékét (*Target class*): másik érték kiválasztásához a legördülő menüben kell másikat választani. A cél-attribútum minden egyes értékéhez a gyakori elempárok rendelődnek hozzá.
  Gyakori elempárok: annak valószínűségét tartalmazza, hogy a cél-attribútum kiválasztott értéke előfordul-e más attribútumok egyes értékeivel. Az első 100 érték alapból megjelenik. Csak akkor tartozik rekord, az attribútum értékéhez a táblázatban, ha az előfordulásának valószínűség és az érték előfordulásának valószínűsége mindkét esetben nagyobb, mint 0.
  Mint azt már korábban tárgyaltuk a Naiv Bayes algoritmus építése közben diszkretizálja az adatokat. Ha a '*Bin*' gombra klikkelünk akkor a diszkretizált adatokat jeleníti meg. Alap beállításnak a diszkretizálatlan adatokat használja.

Tesztelés eredményeinek megtekintése (*Results*)

Az eredményeket megtekinthetjük, egyfelől az elkészült modell '*Result*' fülén a '*Test Metrics*' pont alatt illetve ugyan így elérhetjük, ha a tesztelés szekcióban a szintén '*Result*' menüpontra klikkelünk. Nézzük most sorba a tesztelés kiértékelését.

- A Tesztelés beállításai (*Test Settings*) és a '*Task*' fülön általános információk tekinthetőek meg a beállításokról illetve a bemenetről és az eredményekről.

- Modell tesztelésének értékelése (*Predictive Confidence*) (10. ábra) egy 0 és 1 közé eső szám, amely azt mutatja mennyivel volt jobb az előrejelzése a tesztelésnek, mint a modell készítésének előrejelzése. A naiv modell szám típusú cél-attribútum esetén a középértékét, illetve kategória típusú cél-attribútum esetén pedig annak beállított értékét jelzi előre. A következő formula adja ezt a számot:

$$MAX\left(\left(1-\left(\text{mod}ell\_hibája / naiv\_\text{mod}ell\_hibája\right)\right),0\right)$$

  Kategória típusú cél-attribútum esetén a naiv modell hibája $(n-1)/n$, ahol az n a cél-attribútum különböző értékeinek száma. Például: legyen egy előrejelzés egy kategória típusú cél-attribútum (lehetséges értékei 0 és 1) esetén, melynek eloszlása: 95% - 0 és 5% - 1.

  Ebben az esetben a naiv modell a 0 értéket fogja előre jelezni és 100%-ig helyes lesz, ha a cél-attribútum 0 értéket veszi fel, ellenkező esetben pedig 100%-ig helytelen lesz.

  Átlagosan, ha a cél-attribútum megjósolt értékeit vesszük alapul, akkor az elkészített naiv modell hibás. Ugyanis: $(0\%+100\%)/2=50\%$

  Ha a tesztelés jóságértéke 0-nál nagyobb, akkor a tesztelt modell jobb, mint a naiv modell.

  Numerikus értékek esetén is hasonló elemzést láthatunk: 0 esetén a modell előrejelzése nem jobb, mint a naiv modellé. 1 esetén az előrejelzés tökéletes, és 0.5 esetén a modell előrejelzésének helyessége 50%-kal lecsökken.

- Pontosság (*Accuracy*) Ezen a képernyőn (11. ábra) egy összegzést láthatunk az osztályozó modellről. Opcionálisan megtekinthető a tévesztési mátrix is (*Confusion Matrix*) mely hasznos lehet, ha a cél-attribútum kettőnél több értéket is felvehet. A következők láthatók a fülön: Név (*Name*), Átlagos pontosság: (*Average Accuracy*) Általános Pontosság (*Overall Accuracy*), Teljes költség (*Total Cost*).

  A Modell teljesítményét (*Model Performance)* az alábbi táblázat tartalmazza. Ha pontosabb értékelést szeretnénk megtekinteni, akkor klikkeljünk a '*More Detail…*' gombra. Az oszlopok jelzik az előrejelzéseket, amelyet az osztályozó modell készített. A sorok pedig a mintában lévő aktuális értékeket. A modell általános pontossága 77%-os. 172 olyan eset van melyre sikeres volt az osztályozás abból a szempontból, hogy lojálisak-e a vásárlók, ugyanis ennyi esetben valóban nem azok. És 16 esetben téves volt az

osztályozás ugyanis ők megbízható vásárlók. Hasonlóképpen a második oszlopban 51 lojális vásárlót sorolt rossz csoportba, míg 60 ügyfél esetén jó volt a modell melyet felállított. Azokat az eseteket, amelyeknél rossz volt az osztályozás '*false-negative*' és '*false-positive*' előrejelzéseknek nevezünk, amely a 11. ábrán nagyon jól látható.

- A '*Lift*' fülön két grafikon található. Ezek azt mérik, hogy a véletlenszerű modellekhez képest mennyivel javítja a modell az előrejelzéseket. Ez fontos abból a szempontból, hogy felmérjük a modell hatásfokát. Az egyik diagram az összesítő növekményt (*Cumulative Lift*), a másik pedig az összesített pozitív eseteket (*Cumulative Positive Cases*) ábrázolja. Ezeket a diagramokat szokás nyereség-görbének vagy nyereség-diagramnak is nevezni, és elég népszerű technikák direkt marketing esetén.

  Tegyük fel, hogy készítünk egy osztályozási modellt mellyel szűrni és sorbarendezni tudjuk azokat az ügyfeleket, akik valószínűleg válaszolnak a leveleinkre. Ez a grafikon segítséget nyújt abban, hogy felmérjük, milyen hatékonysággal tudjuk kiválogatni a levelezőlistánk legjavát. Kiszelektáljuk azt a legkevesebb számú ügyfelet, akik a legnagyobb valószínűséggel válaszolnak a leveleinkre. Az ODM a modell által megjósolt adatokat, rendezi a valószínűségek alapján. Majd 10 egyenlő részre vagy osztályra osztja, a tényleges pozitív értékeket.

  A teszt eredmények azt mutatják (12. ábra), hogy ha a felső 40%-ot vagy a 4. részt választjuk, akkor legkevesebb kétszer annyi válasz várható mint egy véletlenszerű mintavételezés esetén.

- ROC analízis (*Receiver Operating Characteristics - ROC*)

  A ROC analízis egy igen hasznos eljárás, hogy értékeljük az osztályozási modellt. A grafikonon a mintavételezés során az aktuális értékek '*false-positive*' és '*true-positive*' értékek százalékos elosztásában láthatóak. A ROC analízis középértékeket határoz meg a modell és a küszöbértékek összehasonlításával, amely a pozitív esetek magas arányát adja. A grafikon bal felső területe az optimális helyeket tartalmazza ahol magas a '*true-positive*' értékek számossága a '*false-positive*' értékekkel szemben.

  A ROC grafikon (13. ábra) segítségével módosíthatóak a modell beállításai és így a tévesztési mátrix változásai megfigyelhetőek. A piros vonal áthelyezésével érhetjük el ezt és ezzel együtt változnak a pozitív előrejelzések illetve mind a költségvetési mind a

tévesztési mátrix. Normál esetben minden egyes esethez tartozik egy előrejelzés, ha ennek a valószínűség 0.5 vagy a felett van, akkor az előrejelzés pozitív.

Minket jobban érdekelnek azok az ügyfelek, akik megbízhatóak, mint azok, akik nem. Szóval, ha mi maximum 100 ügyfélnek szeretnénk brossúrát küldeni, akkor egy részhalmazt kell összeállítanunk. A '*false-negative*'-ok száma az elkészített modellben 51 eset. Ez az érték csökkenthető egészen addig, amíg az összes pozitív előrejelzés 100 alatt marad és a valószínűség 0.5 felett. Így ha 0.522-re állítjuk a valószínűségi küszöbértéket, akkor a '*false-negative*'-ok száma lecsökkent 22-re és a pozitív esetek száma, pedig $41+54=95$ ami kevesebb, mint 100.

Most, hogy rendelkezésünkre áll egy kész modell mellyel meg tudjuk határozni, hogy mely ügyfelek a legmegbízhatóbbak és így nekik akár egy kérdőívet akár egy brossúrát is küldhetünk. Ezek után a modellt felhasználhatjuk, egy adathalmazon melyekről szeretnénk a felállított modell segítségével kideríteni, hogy mégis kik lesznek ezek az ügyfelek. Erről a tevékenységről az Eredmények feldolgozása című részben szeretnék beszámolni.

2. Döntési fák algoritmusa

A Naiv Bayes algoritmus által kapott eredmény, hogy mely ügyfelek megbízhatóak és melyek azok akik kevésbé, nem túl teljes. Ha ennél részletesebb elemzést szeretnénk kapni akkor, nézzük a most következő döntési fákat melyek nem csak egy-egy attribútum közti összefüggést képes felismerni.

A döntési fák alapötlete, hogy bonyolult összefüggéseket egyszerű döntések sorozatára vezet vissza. Egy ismeretlen minta osztályozásakor a fa gyökeréből kiindulva az éleken keresztül a gyermek csomópontokon feltett kérdésekre adott válaszoknak megfelelően addig haladunk lefelé a fában, amíg egy levélelembe nem érünk. A levélelem határozza meg a döntést.

A döntési fák nagy előnye, hogy automatikusan felismerik a lényegtelen változókat. Ha egy változóból nem nyerhető ki információ a magyarázott változóról, akkor azt nem is tesztelik. Ez a tulajdonság azért előnyös, mert így a fák teljesítménye zaj jelenlétében sem romlik. valamint a problémamegértésünket is nagyban segíti, ha megtudjuk, hogy mely változók fontosak és melyek nem. Általában elmondható, hogy a legfontosabb változókat a fa a gyökér közelében teszteli. További előny, hogy a döntési fák nagyméretű adathalmazokra is hatékonyan felépíthetők.

A döntési fák egyik fontos tulajdonsága, hogy egy csomópontnak mennyi gyermeke lehet. Nyilvánvaló, hogy egy olyan fa, amely pontjainak kettőnél több gyermeke is lehet mindig átrajzolható bináris fává. A legtöbb algoritmus ezért csak bináris fát tud előállítani.

A döntési fák előnyös tulajdonsága, hogy a gyökérből egy levélbe vezető út mentén a feltételeket összeolvasva könnyen értelmezhető szabályokat kapunk a döntés meghozatalára, illetve hasonlóan egy laikus számára is érthető módon azt is meg tudjuk magyarázni, hogy a fa miért pont az adott döntést hozta.

A döntési fát a tanító adatbázisból állítjuk elő, úgy hogy olyan kérdést próbálunk kreálni rekurzívan, aminek segítségével az adathalmazt szétvágható. Egy szétvágást akkor tekintünk jónak, ha magyarázandó változó eloszlása a keletkezett részekben kevésbé bizonytalan, mint szétvágás előtt. [9]

Döntési fák ODM-beli használata:

A Naiv Bayes algoritmusnál leírt folyamatot, egészen az algoritmus részletes beállításáig megismételjük, nincs lényeges különbséget a két varázsló közt. Ugyanúgy ahogy a Naiv Bayes algoritmus futását is lehetett konfigurálni úgy ennél az algoritmusnál is van lehetőség erre. A modell építése (*Build*) fül (14. ábra) alatt, a második menüpontnál az algoritmus beállításit (*Algorithm Settings*) lehet pontosítani.

Ez az algoritmus belső optimalizálást használ arra, melyik attribútumot használja az egyes ágak vágásánál. Minden egyes vágáshoz, egy úgy nevezett homogenitási metrikát használ (*Homogeneity Metric*), Két fajta homogenitási metrika közül lehet választani: A ’*Gini*’ az egyik ágat megpróbálja olyan „tisztán” tartani amennyire csak lehet (ez a legmagasabb lehetséges százaléka egy osztálynak), míg az ’*Entropy*’ megpróbálja kiegyensúlyozni az ágakat valamint szétválasztani az osztályokat amennyire csak lehetséges.

A továbbiakban pedig a döntési fa algoritmus megállási feltételeit lehet megadni. Ezek a maximum mélység, minimális rekord egy csomópontban, minimum aránya a bemenő adatoknak egy csomópontban, minimum rekord vágásonként, minimum aránya a bemenő adatoknak egy csomópontban vágásonként.

Miután végeztünk a beállításokkal indítsuk el az algoritmust. Az algoritmus futásának végeztével, nézzük meg az eredményeket, mely eltérő, mint a Naiv Bayes algoritmusban. Először is nézzük meg a modellt, amit készítettünk (15. ábra), láthatjuk, hogy egy döntési fát állított össze az ODM. Hogy részletesebb információt kapjunk az egyes csomópontokra klikkelve, megtekinthetjük, hogy az attribútumokra milyen feltételrendszert sikerült alkotni,

és a tanító mintából hány eset teljesíti azokat. Minden egyes rekord esetén a következő információk állnak rendelkezésünkre:

- Becsült érték (*Predicted Value*): Az adott csomóponthoz tartozó esetek többsége milyen célértékkel rendelkezik.
- Bizonyosság (*Confidence*): Azt fejezi ki, hogy milyen valószínűséggel veszi fel az adott csomóponthoz tartozó eset a becsült értéket.
- Esetek (*Cases*): Az egyes csomópontokhoz tartozó feltételrendszert a mintából hány eset teljesíti.
- Támogatás (*Support*): Százalékos arányban fejezi ki, hogy az egyes csomópontokhoz tartozó feltételrendszert a minta hány százaléka teljesíti.

Eredményként azt láthatjuk, hogy akiknek a családi állapota: házas, és akik képzési szintje: főiskolai diploma, egyetemi diploma, PHD, Professzor, ők 76,06%-os eséllyel tartoznak a megbízható ügyfelek közé. Szóval nekik érdemes kérdőívet küldeni.

Vizsgáljuk meg egy másik oldalról a modellünket. Nézzük meg a ROC analízis eredményét és hasonlítsuk össze a Naiv Bayes modellével. Láthatjuk, hogy ha ezzel a módszerrel szeretnénk a szűrni az ügyfeleinket, akkor látható, hogy ugyan olyan eredmény elérésénél sokkal több, pontosan 80, '*false-negative*' ügyfél található, míg a Naiv Bayes ROC analízisében csak 22. Ezzel a módszerrel nem fogunk előrelépést elérni, viszont más bonyolultabb probléma esetén viszont nagyon jól használható. Például ha egy bank szeretné ügyfeleit szűrni abból a szempontból, hogy kinek milyen hitelkonstrukciókat ajánljon.

3. Szupport vektor gépek (Support Vector Machines - SVM)

Nézzük meg a következő algoritmust, melyet az ODM osztályozási problémák megoldásra nyújt. A számos alkalmazási területen jó eredményeket adó szupport vektor gépekkel történő osztályozás esetén is az egyik leghatékonyabb módszerek egyike. Az SVM alapverziója az bináris osztályozási problémák megoldására alkalmas. A következő rész megértéséhez nézzük először a hipersík definícióját:

Legyen $c'x = c_1x_1 + ... + c_nx_n = b$, $c' \neq 0'$ egy n változós lineáris egyenlet $c_1, c_2, ..., c_n$ ismert számértékek, és $X = \{x \mid c'x = b\}$ azon pontok halmaza, amelyek kielégítik a fenti egyenletet.

Ez a halmaz az n dimenziós euklideszi tér egy hipersíkja.

A többi lineáris osztályozóhoz képest az a fő ismérve, hogy nemcsak egyszerűen egy olyan hipersíkot keres, amely elválasztja a pozitív és negatív tanítómintákat, hanem ezek közül a

legjobbat, vagyis intuitíve azt, amelyik a két osztály mintái között épp „középen” fekszik. Másképpen fogalmazva olyan döntési hipersíkot határoz meg, amely maximalizálja a hipersík és a legközelebbi pozitív és negatív tanítóadatok közti eltérést. Ezeket a tanítóadatokat szupportvektoroknak nevezzük. A hipersík meghatározásában a tanítóadatok közül csak a szupportvektorok játszanak szerepet.

Ha megnézzük miért előnyös, ha ezzel a módszerrel adjuk meg a szeparáló hipersíkot:

- Egyrészt azért, mert a döntési felülethez közeli pontok osztályba-sorolása a legbizonytalanabb. Minél kevesebb pont esik erre a területre, annál kevesebb bizonytalan döntést hoz az osztályozó.
- Másrészt a maximális pozitív és negatív tanítóadatok közti eltérés által meghatározott szélességű szeparáló sáv elhelyezésére sokkal kevesebb lehetőség van, mint egy tetszőleges szeparáló hipersík esetén. Ez azt jelenti, hogy kevésbé függ a konkrét adatoktól, ezért nagyobb általánosító képességgel bír az osztályozási modell. [2]

Az SVM algoritmus eltér az eddig bemutatott eljárásoktól, ugyanis különböző típusú adatokra is nagyon jól használható. SVM alkalmazható regressziós problémák megoldására is, ahol az előrejelzett értékek folytonosak, ellentétben a diszkrét típusú adatokkal.

Egy másik nagy előnye az SVM algoritmusnak, az az, hogy szöveges eredményeket ad. Ha például rendelkezésünkre áll egy kórház betegeinek orvosi adati vagy egy olyan felmérés eredménye mely az ügyfelek elégedettségét méri vagy más szöveges információ, ezen adatok mind felhasználhatóak az SVM bemeneteként.

SVM algoritmus ODM-beli használata:

A Naiv Bayes algoritmusnál felvetett problémára állítsunk fel most egy modellt az SVM algoritmussal. Az eddig megszokott módon végrehajtjuk azokat a lépéseket, melyeket a Naiv Bayes algoritmusnál majd mikor a legutolsó lépéshez érünk, pontosítjuk az algoritmus beállításait. Hasonló képpen itt is van lehetőség mintavételezésre. A mintavételezést itt is ajánlott használni, ha csökkenteni szeretnénk a futási időn. A megszokottakhoz képest két újabb transzformáció beállítást teszi kötelezővé a varázsló. Egyrészről a kilógó adatok (*Outlier Treatment*) és a hiányzó értékek (*Missing Values*) kezelését. Az SVM érzékeny ezen fajta értékekre. Alap esetben a numerikus típusú hiányzó értékeket az adott attribútum értékhalmazának középértékére, míg a kategória típusú értékeket a leggyakrabban előforduló értékre cseréli. Persze definiálható más, konkrét érték is.

Az SVM algoritmus esetében szükséges a numerikus típusú adatok normalizálása. Alapértelmezetten az adatok egy 0 és 1 intervallumba normalizálja, a másik módszer a z-érték módszer, mely esetében az értékek a -1 és 1 intervallumból kell, hogy kikerüljenek.

A modell készítése fülön az algoritmust futásának mag függvényét (*Kernel function*) lehet beállítani. Az N attribútummal rendelkező adathalmaz tekinthető egy N dimenziós térben elhelyezkedő pontok halmazának is. Az SVM ezt a halmazt próbálja meg hipersíkokkal szeparálni, erre két mód közül lehet választani:

Nem lineáris – Gauss esetben (*Gaussian*): osztályozás és anomáliák felfedezése esetén ezt a módot érdemes használni, ahol az osztályok nem-lineárisan szeparálhatók. Vagyis az osztályok nem választhatók külön egyenessel vagy síkkal.

Lineáris esetben (*Linear*): A modell építésére használt minden egyes attribútum együtthatója rangsorolva lesz, így látható, hogy melyek játszottak leginkább szerepet a célérték meghatározásában.

Ha lefuttatjuk mind a két módszerrel a modell készítését, akkor a következőket eredményeket kapjuk (17. ábra és 18. ábra):

		Gauss mód	Lineáris mód
Modell tesztelésének értékelése		59,43%	57,59%
Pontosság	*false-negative*	338	327
	false-positive	26	25
	true-negative	103	114
	true-positive	125	126
Átlagos pontosság		79,71%	78,77%
Általános pontosság		78,20%	76,52%

Osztályozás probléma esetén valóban a Gauss módszerrel kapott eredmények egy kicsivel jobbak. A Gauss mód esetén több esetet sikerült jól osztályozni és kevesebbet rossz osztályba sorolni. Ez látszik az átlagos és az általános pontosság eredményein is.

Majd ha ezek után megnézzük meg a ROC analízissel kapott eredményeket (19. ábra és 20. ábra), akkor láthatjuk, hogy:

Sokkal jobb modellt sikerült alkotni SVM algoritmussal, mint a Naiv Bayes algoritmussal.

Másodszor a valószínűségekből is jól látható, hogy ha visszatérünk az eredeti kérdésre akkor sokkal több megbízható ügyfelet sikerül behatárolni ezzel a modellel. Nagyobb számban fordulnak elő helyesen osztályozott esetek. Így nagyobb valószínűséggel sikerül az ügyfelek

közül azokat kiválogatni, akik majd válaszolnak a kérdőívre, vagy érdemes nekik brossúrát küldeni.

Regressziós algoritmusok (*Regression*)

A regressziószámítással azt vizsgálhatjuk, hogy egy változó értéke (függő változó) hogyan közelíthető, jósolható meg más változók (független változók) segítségével. A lineáris regressziónál az adatok egy egyenes vonal segítségével modellezzük. A regressziós modell tulajdonságai alapján megkülönböztethetünk lineáris és nemlineáris regressziót. A lineáris regresszió a regresszió legegyszerűbb formája. A kétváltozós lineáris regresszió egy Y valószínűségi változót (függő) modellezi egy másik X valószínűségi változóval (független). Azaz $Y = \alpha + \beta X + \varepsilon$, ahol az Y szórását állandónak vesszük, az α és β regressziós együtthatók pedig sorra megadják az egyenes y tengellyel vett metszetét és meredekségét és az ε a hibatag. Ezek az együtthatók több módszerrel is kiszámíthatók, például legkisebb négyzetek módszerével. A nem-lineáris regresszió lényege egy egyenlet illesztése az adatok értékeihez és annak a vizsgálata, hogy az adatok illeszkednek-e az egyenlet által meghatározott görbéhez.

Az ODM két algoritmust nyújt regressziós problémák megoldására, a lineáris regressziót (*Linear Regression*) és az SVM algoritmust. A most következő részben bemutatom az SVM algoritmus működését regressziós probléma esetén.

1. Szupport vektor gépek (Support Vector Machines - SVM)

Az SVM algoritmus folytonos értékek (lebegőpontos típusú értékek) előrejelzésére használható. Ilyen esetben regressziós problémáról beszélünk. Eme módszer bemutatásához az eddig használt adattáblát fogjuk felhasználni, viszont cél-attribútumként egy folytonos típust választunk, mégpedig az ügyfelek életkorát (*AGE*). Az eddig használt, modell építése varázsló segítségével készítjük el a modellt. Miután kiválasztottuk az regressziós funkciókat majd az SVM algoritmust lépjünk is a következő oldalra. Cél-attribútum az életkor lesz.

A következő lépések ugyan azok, mint a Naiv Bayes algoritmusnál. Viszont az utolsó lépésnél, konfiguráljuk az algoritmus pontos működését. Lehetőség van az alapértelmezett beállításokat módosítani, beleértve az SVM futásának módját is (Lineáris vagy Gauss), az egyetlen különbség, az osztályozáshoz képest, az epszilon értékek beállítása. Az SVM egy úgynevezett epszilon-érzéketlen veszteség függvényt (*epsilon-insensitive loss function*) használ, regressziós probléma megoldásakor. Az SVM regressziós probléma esetén megpróbálja megtalálni azt a folytonos függvényt, melyre a maximális számú adatpont esik az

epszilon-sugarú érzéketlenségi körbe. Azon előrejelzések, melyek epszilon távolságra esnek a valós célértéktől nem értelmezhetőek hibaként. Az epszilon tényező egy szabályozási beállítás SVM regresszió esetén. Ez szabályozza a modell megbízhatóságának hibatűrését, hogy az új adatokra a lehető legjobb általánosítást kaphassuk. Az SVM megkülönböztet kisebb és nagyobb hibákat, a kettő közti különbséget határozza meg az epszilon érték. Az algoritmus belsőleg számolja ki és optimalizálja ezt az értéket vagy beállíthatjuk. Ha a kategória típusú értékek számossága nagyon magas, akkor megpróbálja az epszilont csökkenteni, egy előzetes, a rendszer által kikalkulált értékből.

Miután ezt befejeztük, készítsük el a modellt és nézzük meg milyen eredményt kapunk (21. ábra). Láthatjuk, hogy mindazok a lépések, melyeket beállítottunk egymásután lefutnak.

Először a modell eredményét tekintsük meg (22. ábra). Az alapbeállításoknál láthatjuk, hogy az algoritmus melyik futási módot választotta, hogyha ezt a döntést ráhagytuk, és nem specifikáltuk. Ugyancsak itt találjuk meg a kiválasztott cél-attribútumot és még más információkat.

Most tekintsük meg a tesztelés eredményét (23. ábra). A becslések pontosságának és pontatlanságának meghatározására számos lehetőség kínálkozik. Egyes lehetőségek egy számmal jellemzik a független és a függő paraméterek közti különbséget. A matematikai statisztikából is ismert hibaszámítási módszert kínál az ODM a modellek által becsült paraméter és a független változó tényleges értéke közti különbség jellemzésére. A két lehetőség, számos megközelítést jelent, melyek közül nem lehet egyértelműen kijelenteni, hogy valamelyik módszer a legjobbnak tekinthető. A módszerek bemutatására jelöljük y_i-vel a függő változó tényleges értékét az *i*-edik objektum esetén, $\bar{y}_i$-vel a függő változó becsült értékét az *i*-edik objektum esetén.

A következő statisztikákat vezethetjük be a becslés hibájának mérésére: [11]

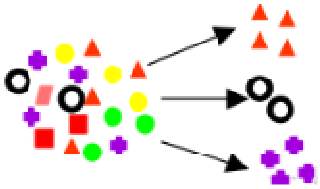
Mean Absolute Error: $MAE = \left(\frac{1}{n}\right)\sum_{i=1}^{n}\left|y_i - \bar{y}_i\right|$

Root Main Square Error: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}_i\right)^2}$

A következő eredmény, amit megtekinthetünk az a regressziós modell grafikus ábrázolása. (24. ábra) A '*residual plot*' az adatok szórásának ábrázolása ahol az *x*-tengely *x* értékének előrejelzése, és az *y*-tengely *x* maradéka. A maradék az *x* aktuális értéke és *x* előrejelzett értékének különbsége. Ha az egér mutatójával rámegyünk egy konkrét értékre, akkor több

információt is megkaphatunk, például a konkrét értéket, koordinátát, előrejelzés valószínűségét. Illetve a pontos értékeket megtekinthetjük, ha az alsó fülön átváltunk a következőre.(25. ábra).

Csoportosítási algoritmusok (*Clustering*)



A klaszterezés egy adathalmaz pontjainak, rekordjainak hasonlóság alapján való csoportosítását jelenti. A klaszterezés szinte minden nagyméretű adathalmaz leíró modellezésére alkalmas. Az adatok klaszterezése bevett technika mind az adatbányászatban mind más például a statisztika, bioinformatika, mintafelismerés, és gépi tanulás területén. Csoportosíthatunk weboldalakat tartalmuk, webfelhasználókat böngészési szokásaik, kommunikációs és szociális hálózatok pontjait közösségeik, kémiai vegyületeket szerkezetük, géneket funkcióik, betegségeket tüneteik szerint.

A klaszterezéshez hasonló feladat az osztályozás. A klaszterezéstől az osztályozási feladatot az különbözteti meg, hogy az osztályozás felügyelt, a klaszterezés pedig nem felügyelt csoportosítás. Osztályozás esetén a választható osztályok valamilyen ábrázolással, mintával vagy modellel leírva előre adottak. Klaszterezés esetén nincs előzetesen megadott cél-attribútum és célérték, az adatok maguk alakítják ki a klasztereket és azok határait.

Az előző fejezetekben foglalkoztunk felügyelet melletti osztályozással. Azaz kiválasztottuk a cél-attribútumot, illetve az előrejelzés pontosságát, melynek következményeképpen előállt, hogy hány esetben osztályozódtak helyesen az adatok a célértéket tekintve. Azon klaszterező algoritmusok esetében ahol a cél-attribútum meghatározása elmarad, azoknál egyszerűen nyomonkövethetjük milyen mintát követ ez a technika.

Klasztereket különböztethetünk meg például, nagyszámú kórházi páciens tekintetében, melyek hasonló jellemvonásokat mutatnak, mint szívbetegek, vagy gyerekgyógyászati páciensek stb. Valamint konkrét rákbetegek, akik egy konkrét gén hiba által okozott tumor válfajában szenvednek. Ők érzékenyek lehetnek bizonyos gyógyszeres kezelésekre. A klaszterezés érdekes összefüggéseket mutathat ki gyógyszerek, gének és betegségek között valamint hogy hogyan reagálhatnak legjobban bizonyos terápiákra.

Oracle Data Mining két algoritmussal rendelkezik arra nézve, hogy hogyan hajtsunk végre klaszter analízist. A K-középpontú klaszteranalízis illetve az Ortogonális partícionáló klaszterezés (O-Cluster). Ez után nézzük a következő algoritmusokat.[1]

1. K-középpontú klaszteranalízis

A K-középpontú algoritmus véletlenszerű kezdeti súlypontot (*centroid*) definiál, amelyek valójában gravitációs középpontnak minősülnek, majd maga az algoritmus távolság mérést használ, hogy kiszámolja a súlypont és az adat objektumok távolságát. Ehhez a távolságmérés lehet euklideszi vagy koszinusz távolságmérés.

A K-középpontú klaszteranalízissel is a vizsgált elemek klaszterbe történő besorolását végezhetjük el. A módszer főbb lépései a következők:

- Adjuk meg a kezdeti klaszterek számát.
- Minden elemet besorolunk a hozzá legközelebb eső klaszterbe.
- Határozzuk meg a klaszter-centroidokat.
- Az elemeket átsoroljuk úgy, hogy a csökkenjen az elemek és a centroidok közötti távolság
- A 3-4. lépés ismétlése addig, amíg a klaszterek nem állandósulnak. [10]

A K-középpontú klaszteranalízis ODM-beli használata:

K-középpontú algoritmus a legismertebb általános metódus és sok helyen alkalmazzák. Az adatbányász csak meghatározza a számosságot, hogy hány klaszter keletkezzen, így csökkenti az esélyét, hogy problémát okozzon az eredmény klaszterbe csoportosítása. Amikor az adatok nagymértékben különböző méretű vagy szokatlan formájú (nem gömb alakú csoportosítású) klasztereket tartalmaznak, avagy sok kilógó adatponttal valamint véletlenszerű eseményekkel (zaj) rendelkeznek, akkor a K-középpontú algoritmusnak gondjai lesznek a klaszterek feltérképezésével.

Ez az algoritmus hasonlóan az osztályozó algoritmusokhoz, a modell készítése menüpontból érhető el. Itt az osztályozás helyett a klaszterezést és azon belül a K-középpontú klaszteranalízist kell választani. Majd kiválasztjuk, azt az adattáblát, melyet klaszterezni szeretnénk és azok adattagjait.

A futtatás után tekintsük meg az eredményeket. Az első fülön (26. ábra) láthatjuk a létrehozott klasztereket. A végleges klaszterek a fa struktúra levélelemi tartalmazzák. Az egyes klaszterekre vonatkozó feltételrendszer a szabályok (*Rules*) fülön (27. ábra) láthatóak.

Egy klaszter kijelölése után nézzük meg annak részleteit (*Detail*). Egy hisztogram mutatja az egyes attribútum eloszlásokat melyek a kiválasztott klaszterre vonatkoznak. Egyszerre több klaszter hisztogramja is megtekinthető, így könnyen összehasonlíthatjuk a klaszterbe sorolt jellemvonásokat.

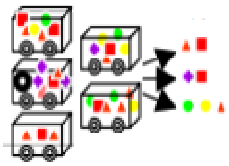
2. Ortogonális partícionáló klaszterezés – O-Cluster

Az *O-Cluster* hierarchikus rács alapú klaszterező modellt hoz létre, amely párhuzamos tengelyű ortogonális partíciókat hoz létre a bemeneti attribútumokból. Az algoritmus rekurzívan működik. A végeredményül szolgáló hierarchikus struktúra a következő formában mutatkozik meg: egy szabálytalan rács, amely kis kockákból építi fel a klasztereket, amely kockák lefedik az attribútumokat. Az eredményül szolgáló klaszter sűrűségi területeket definiál az adat attribútumaira nézve. A klasztereket az attribútum tengelyek mentén definiált intervallumok a hozzájuk tartozó centroidok és hisztogramok írják le. Az érzékenységnek (*sensitivity*) hívott paraméter határozza meg a sűrűségi szint alapját (*baseline*). A klaszterek azonosításakor csak olyan területeket vesz számításba ahol a sűrűségi csúcsok az alapvonal fülé esnek.

A K-középpontú algoritmus olyan attribútum értékeket is képes kockákból felépíteni ahol más természetes klaszterek nem létezhetnek, például ha van, egy olyan terület ahol egységes a sűrűség a K-középpontú algoritmus ezt maximum n darab klaszterré osztja fel ahol n a felhasználó által definiált érték, azaz az algoritmus nem hoz létre kötelezően annyi klasztert amennyi a megadott érték volt, így az egyes klaszterekhez való tartozás tisztábban definiálható. Az O-cluster elkülönít magas sűrűségű területeket az alacsony sűrűségűektől az által, hogy határoló síkokat definiál. Mégpedig úgy hogy megkeresi az attribútumok sűrűégi eloszlási görbéjében található csúcspontok közötti völgyeket. Egy alacsonyabb érzékenységi érték mélyebb völgyeket követel, meg míg a magas érzékenység sekélyebb völgyeket is felfedezi. Így a magasabb érzékenységi érték magasabb számú klasztert eredményez. Az O-Cluster - nek polinom hisztogramok kellene (konkáv és konvex). Ha egy területnek vannak vetületei egységes vagy monoton növekvő sűrűséggel az O-Cluster nem képes ezt felosztani. Amennyiben a klaszter létrehozás nagyon lassú a maximum buffer méret növelésével javíthatunk a teljesítményen.

Az O-Cluster által létrehozott klasztereket Bayes-szűrés valószínűségi modell generálására szokás alkalmazni. Amelyet aztán a modell feldolgozása során használhatunk arra, hogy adatpontokat rendeljünk a klaszterekhez. A generált valószínűségi modell egy kevert modell ahol a komponensek keverékét a számszerű attribútumokra számított független normál eloszlások eredménye, illetve a kategória típusú attribútumok polinom szórása mutatja be.[10]

Asszociációs algoritmusok (Association Rules)



Az asszociációs analízis az adatbázisban tárolt rekordok közötti összefüggések feltárását szolgálja. Leggyakoribb alkalmazási területe a vásárlói kosarak meghatározása. Ez lehetőséget teremt a kereskedők számára azon termékcsoportok összeállítására, melyet a vásárlók sok esetben azonos időben vásárolnak meg.

Az asszociációs analízis a következő típusú szabályokat generálja: „A lisztet vásárlók 80%-a tojást is vett". A példában szereplő 80%, a szabály konfidencia faktora. A másik jellemző érték, hogy az adatbázis rekordjainak hány százaléka tartalmazza az adott termékeket (esetünkben a tojást és a lisztet), ezt a szabályt támogatási szintjének (*support level*) nevezzük. Általában elmondható, hogy azok a szabályok fontosabbak, melyeknek magas a konfidencia faktoruk és a támogatási szintjük, a kevésbé nyilvánvaló összefüggésekre azonban gyakran az alacsonyabb értékekkel rendelkező szabályok alapján következtethetünk.

Alkalmazási területei közül érdemes megemlíteni az adatok tisztítását, azon változók kiszűrését, melyek azonos információt tartalmaznak (pl. életkor és születési dátum).

Az asszociációs analízissel szemben támasztott legfontosabb követelmény, hogy nagyon nagy méretű adathalmazt is képes legyen kezelni, hiszen a jelentősebb kereskedők százezres nagyságrendű termékszámmal és több milliós ügyfélkörrel rendelkezhetnek.

1. Apriori algoritmus

Az apriori algoritmus a gyakori elemhalmazok meghatározására használható algoritmus. Szélességi bejárást alkalmazva, ami azt jelenti, hogy a legkisebb mintából kiindulva szintenként halad előre a nagyobb méretű gyakori elemhalmazok meghatározásával. A következő szinten az eggyel nagyobb méretű elemhalmazokkal foglalkozik. Az iterációk száma legfeljebb eggyel lehet több, mint a legnagyobb gyakori elemhalmaz mérete.

A jelöltek definiálásánál a következő tényt használja fel. Gyakori elemhalmaz minden részhalmaza gyakori. Az állítást indirekten nézve elmondhatjuk, hogy egy elemhalmaz biztosan nem gyakori, ha van ritka részhalmaza. Az apriori algoritmus ezért építkezik lentről. Egy adott iterációban csak olyan jelöltet veszünk fel, amelynek összes valódi részhalmazáról tudjuk, hogy gyakori.

Az algoritmus onnan kapta a nevét hogy az l-elemű jelöltek halmazát J_l-lel, az l-elemű gyakori elemhalmazokat pedig GY_l-lel jelöljük.

A kezdeti érték beállítása után egy ciklus következik, amely akkor ér véget, ha nincsen egyetlen l-elemű jelölt sem. A cikluson belül először meghatározzuk a jelöltek támogatottságát. Ehhez egyesével vesszük a tranzakciókat, és azon jelöltek számlálóját növeljük eggyel, amelyeket tartalmaz a vizsgált tranzakció. Ha rendelkezésre állnak a támogatottságok, akkor a jelöltek közül kiválogatjuk a gyakoriakat.[9]

Apriori algoritmus ODM-beli használata.

Az Oracle hivatalos ODM kézikönyve alapján fogom bemutatni az Apriori algoritmus működését. (Online elérhetőség hivatkozása: Irodalomjegyzék [7], pontosabban annak 13. fejezete és A, B függeléke).

Használjuk most a jól ismert modell építése varázslót egy az Oracle példa adatbázisán (*SH.SALES* és *SH.PRODUCTS*). A '*SALES*' tábla tartalmazza az egyes eladással kapcsolatos információkat, illetve a '*PRODUCTS*' tábla pedig az eladható termékek listáját. A tranzakciókat tartalmazó táblát és a termékek azonosítóját tartalmazó oszlopot be kell állítani. Lehetőség van még egy leíró tábla megadására. Egyszerűen csak a táblát kell megadni, annak a külső kulcsát és azt az oszlopot, amelyik az egyes termékek nevét tartalmazza (28. ábra).

Ha lefutott az algoritmus, akkor az eredményeknél láthatjuk, hogy mekkora valószínűséggel és mely feltételek mellett vásároltak bizonyos termékeket (29. ábra). A felállított szabályokra lehetőség van szűrési feltételeket beállítani. Nézzük meg milyen feltételek mellett vásároltak például egér alátétet. Állítsuk be a következmény (*consequent*) oldalon az egér padot és állítsuk be a feltételezés valószínűségét 80%-ra (30. ábra). Ez által azok azokat a termékeket kapjuk meg melyeket elég nagy valószínűséggel egér pad mellé vásároltak (31. ábra).

Nagyon könnyedén és rugalmasan használható ez az algoritmus, kereskedők számára, akik növelni szeretnék a bevételüket az által, hogy termékeiket megfelelően csoportosítják.

Eredmények feldolgozása

Miután elkészítettük a modellünket, leteszteltük és beállítottuk, hogy az üzleti kívánalmaknak megfelelően működjön, akkor ezt a modellt felhasználhatjuk arra, hogy megvizsgáljuk ügyfélkörünkből, kik azok a személyek akik eleget tesznek a modell által támasztott követelményeknek.

Az ODM tartalmaz egy varázslót mely sokban segíti a felhasználót eme a fázis kivitelezésében. Sőt mi több ez a varázsló zökkenőmentesen végigvezeti a felhasználót a modell alkalmazásának (*Apply*) folyamatán.

Az SVM által létrehozott modellt, használjuk fel, ugyanis a három osztályozási algoritmus közül ennek volt a legjobb az előrejelzési valószínűsége. Először is a varázslót az '*Activity* → *Apply*' menüpont megnyitásával indíthatjuk el. Majd válasszuk ki, melyik modellt szeretnénk felhasználni. Az ODM kilistázza az összes eddig létrehozott modellt. A harmadik lépésben ki kell választani azt az adattáblát, amelyen szeretnénk a modellt használni (32. ábra). Láthatjuk, hogy kik azok az ügyfelek akik a legnagyobb valószínűséggel válaszolnának a leveleinkre. Befejezve a műveletet, tekintsük meg a modell használatával kapott eredményt (33. ábra). Az eredmények egy táblában tárolódnak, így nagyon könnyen írhatunk rá egy programot mely a megadott táblában lévő személyeknek azonosító alapján küldjön levelet majd a visszaérkező választ dolgozza fel. Erre van lehetőségünk például az Oracle Java API – on keresztül.

Amikor egy modell, mely számunkra a legjobb megoldást nyújtja, elkészül, akkor az adatbányászat eredményét, amely a modell használatával érhető el, valamilyen üzleti cél megvalósítására felhasználhatjuk. Az eredményeket több módon is exportálhatjuk az ODM-ből:

- Kimenthetjük szöveges állományként vagy *xls* fájlként.
- Az *Oracle Discoverer* program segítségével publikálhatjuk.
- Exportálhatjuk a modell egy eredményét egy másik Oracle adatbázis szerverbe.

Természetesen a szöveges formátumba exportált eredmények nem túl látványosak, de például: Amennyiben *Excel* formátumot választottunk akkor egyszerűen használhatjuk annak beépített lehetőségeit, grafikon, hisztogramok és egyéb kimutatások grafikus megjelenítésére.

Exportálás helyett, mivel a végeredmény már adatbázisba eltárolásra került, használhatunk analitikus eszközöket melyek, könnyedén csatlakoznak adatbázisokhoz és lehetőségek tárházát biztosítja számunkra a látványos, kifejező és lényegretörő prezentációkat.

Összefoglalás

Sok hasznos tapasztalatra tettem szert az adatbányászati technikák felfedezése során. Mialatt diplomamunkámon dolgoztam több osztályozási, regressziós és klaszterező algoritmust megismertem és használtam, például a szupport vektor gépek melyek gyakorlati jelentősége elég nagy.

Idő hiányában sajnos nem sikerült az általam kitűzött célokat megvalósítani, mivel könnyedén el lehet veszni a részletekben. Minél mélyebbre ástam magam, annál inkább kezdtem ráébredni ennek a témának a sokrétűségére. Az egyes algoritmusok megértése és bemutatása sokkal több időt vesz igénybe. Mivel számomra az adatbányászat rendkívül izgalmas témakör ezért szándékomban áll kamatoztatni a most megszerzett tudást és továbbiakban is figyelemmel kísérni ennek a nagyszerű technológiának a fejlődését.

Funkcionalitás szempontjából közel sem sikerült az ODM összes adatbányászati algoritmusát bemutatni. Számos egyéb modul is kötődik ehhez a termékhez. Itt van például a Java API, melynek segítségével nagyon könnyedén lehet üzleti igényekre szabott alkalmazást írni mellyel megkönnyíthető az adatbányászati folyamat. Másfelől adott a PL/SQL könyvtár, amely egyszerű átjárhatóságot kölcsönöz az adatbázis és az ODM között. A modell PL/SQL kód formájában történő előállítása azzal az előnnyel jár, hogy a továbbiakban nincs szükség az ODM futtatási környezetre. A kódot egyszerűen alkalmazhatjuk bármely más általunk fejlesztett alkalmazásban és kiaknázhatjuk annak előnyeit.

Itt ragadnám meg az alkalmat, hogy köszöntet mondjak Dr. Ispány Márton Tanár Úrnak a diplomadolgozat elkészítésében nyújtott segítségért és ötletekért.

Köszönöm testvéremnek Szabó Tamásnak, aki fáradhatatlan kitartásával segítette ennek a munkának a megalkotását.

Végezetül külön köszönöm barátnőmnek, Varga Boglárkának, aki délutánonként megteremtette azt a nyugodt környezetet, amely lehetőséget adott, hogy a mindennapi feladatok mellett a diplomámmal foglalkozhassak.

Irodalomjegyzék

[1]. Oracle® Data Mining [könyv]

Szerző: Dr. Carolyn K. Hamm,

Kiadó: Kittrel, Észak Carolina, USA: Rampant TechPress, 2007

[2]. Szövegbányászat [könyv]

Szerző: Tikk Domonkos,

Kiadó: Budapest, Typotex Elektronikus Kiadó Kft, 2006

[3]. Adatbányászat [könyv]

Szerző: Jiawei Han és Micheline Kamber

Kiadó: Budapest Panem Kft, 2004

[4]. CRISP-DM - Process Modell [Online]

http://www.crisp-dm.org/Process/index.htm

[5]. Data Mining Glossary

http://www.twocrows.com/glossary.htm

[6]. Oracle® Data Mining - Wikipedia [Online]

http://en.wikipedia.org/wiki/Oracle_Data_Mining

[7]. Oracle® Data Miner Classic Tutorial [Online]

http://www.oracle.com/technology/products/bi/odm/odminer.html

[8]. Oracle® Data Mining Application Developer's Guide [Online]

http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28131/toc.htm

[9]. Adatbányászati algoritmusok [Online]

http://www.cs.bme.hu/~bodon/magyar/adatbanyaszat/tanulmany/adatbanyaszat.pdf

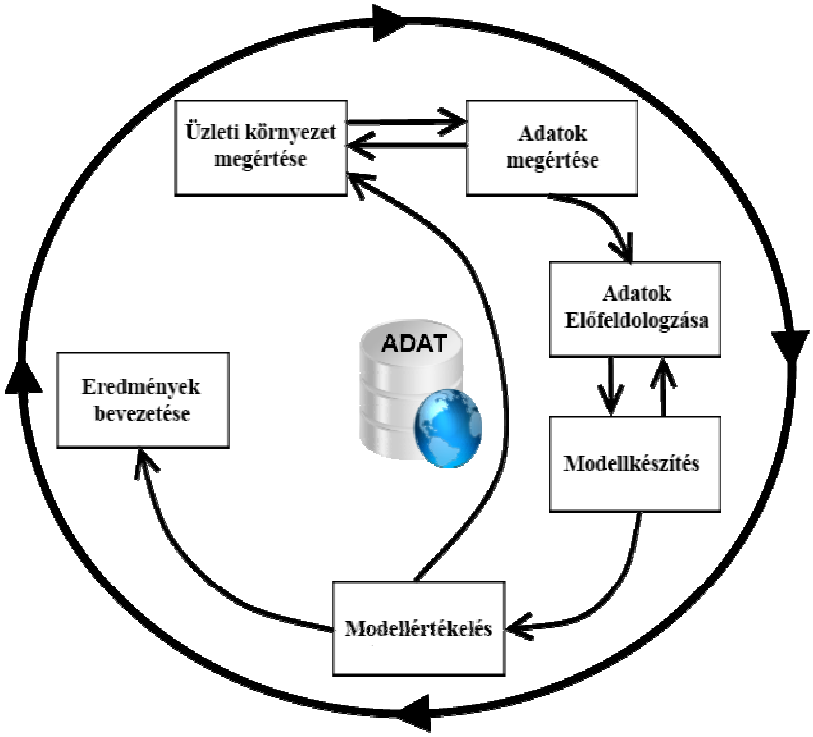
[10]. Oracle® Data Mining Concepts, O-Cluster [Online]

http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/algo_oc.htm

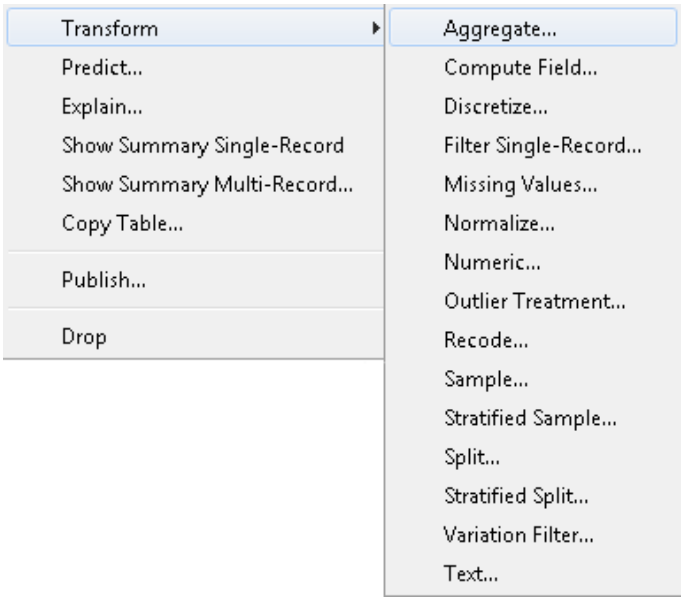
[11]. Oracle® Data Mining Concepts, Regression [Online]

http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/regress.htm

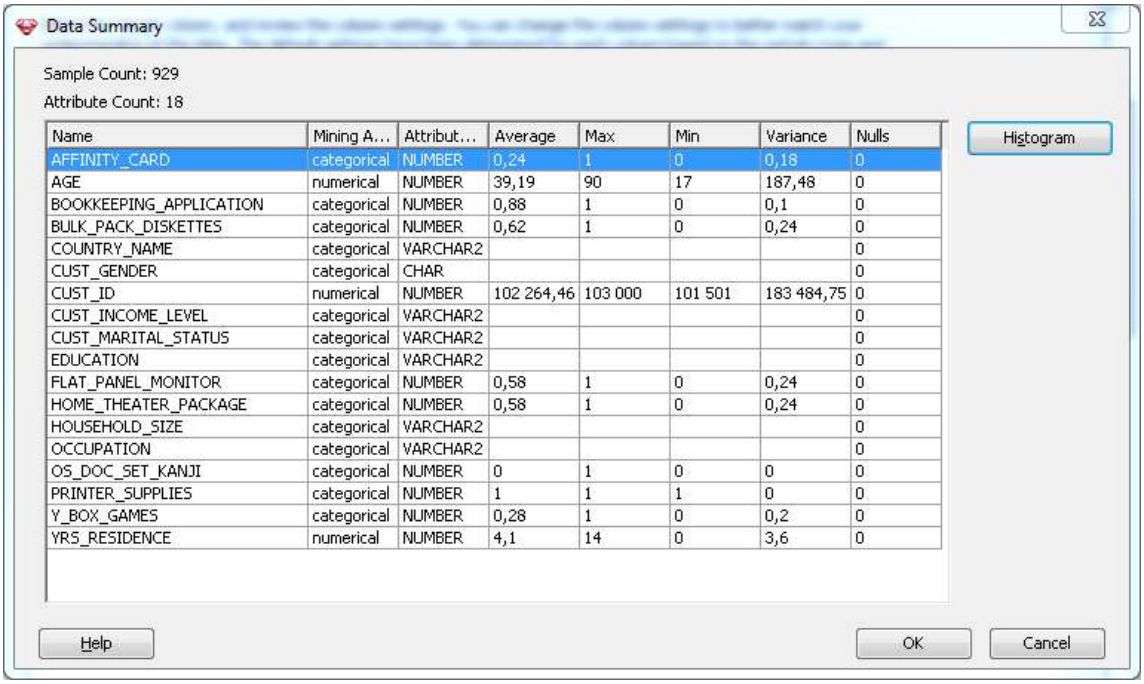
Függelék



1. CRISP-DM diagram, adatbányászat életciklusa.

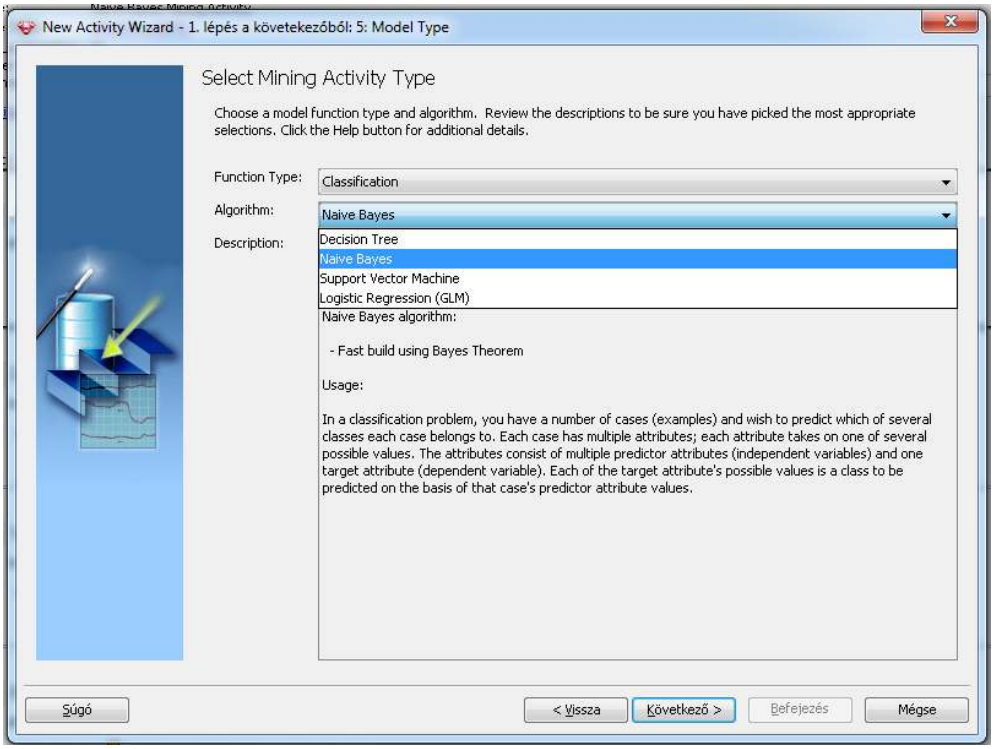


2. ODM - ben elérhető transzformációk.

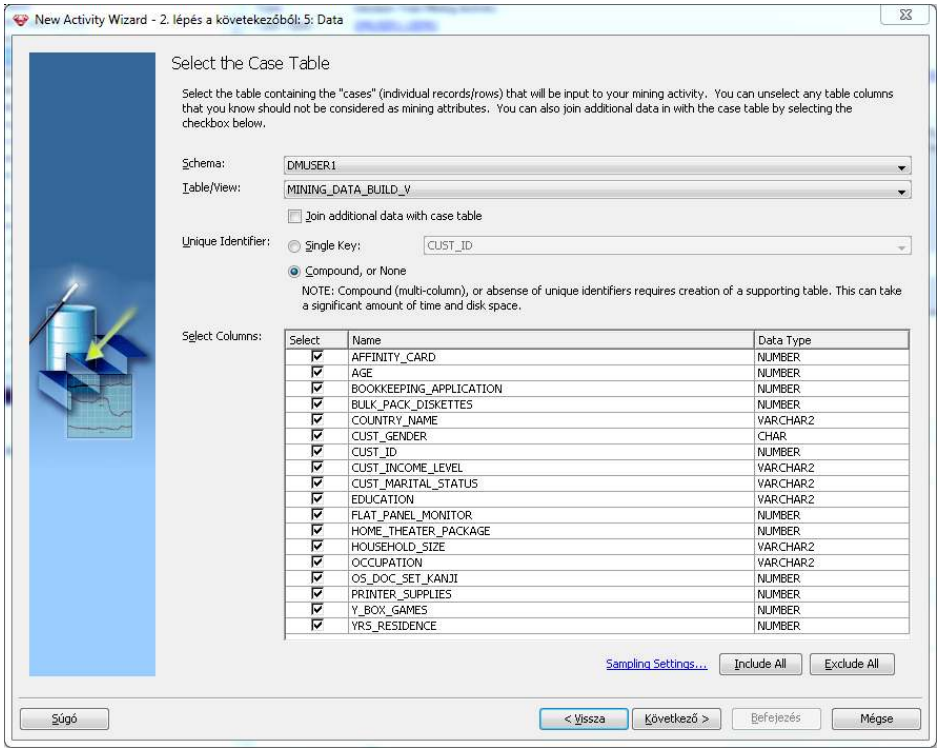


Name	Mining A...	Attribut...	Average	Max	Min	Variance	Nulls
AFFINITY_CARD	categorical	NUMBER	0,24	1	0	0,18	0
AGE	numerical	NUMBER	39,19	90	17	187,48	0
BOOKKEEPING_APPLICATION	categorical	NUMBER	0,88	1	0	0,1	0
BULK_PACK_DISKETTES	categorical	NUMBER	0,62	1	0	0,24	0
COUNTRY_NAME	categorical	VARCHAR2					0
CUST_GENDER	categorical	CHAR					0
CUST_ID	numerical	NUMBER	102 264,46	103 000	101 501	183 484,75	0
CUST_INCOME_LEVEL	categorical	VARCHAR2					0
CUST_MARITAL_STATUS	categorical	VARCHAR2					0
EDUCATION	categorical	VARCHAR2					0
FLAT_PANEL_MONITOR	categorical	NUMBER	0,58	1	0	0,24	0
HOME_THEATER_PACKAGE	categorical	NUMBER	0,58	1	0	0,24	0
HOUSEHOLD_SIZE	categorical	VARCHAR2					0
OCCUPATION	categorical	VARCHAR2					0
OS_DOC_SET_KANJI	categorical	NUMBER	0	1	0	0	0
PRINTER_SUPPLIES	categorical	NUMBER	1	1	1	0	0
Y_BOX_GAMES	categorical	NUMBER	0,28	1	0	0,2	0
YRS_RESIDENCE	numerical	NUMBER	4,1	14	0	3,6	0

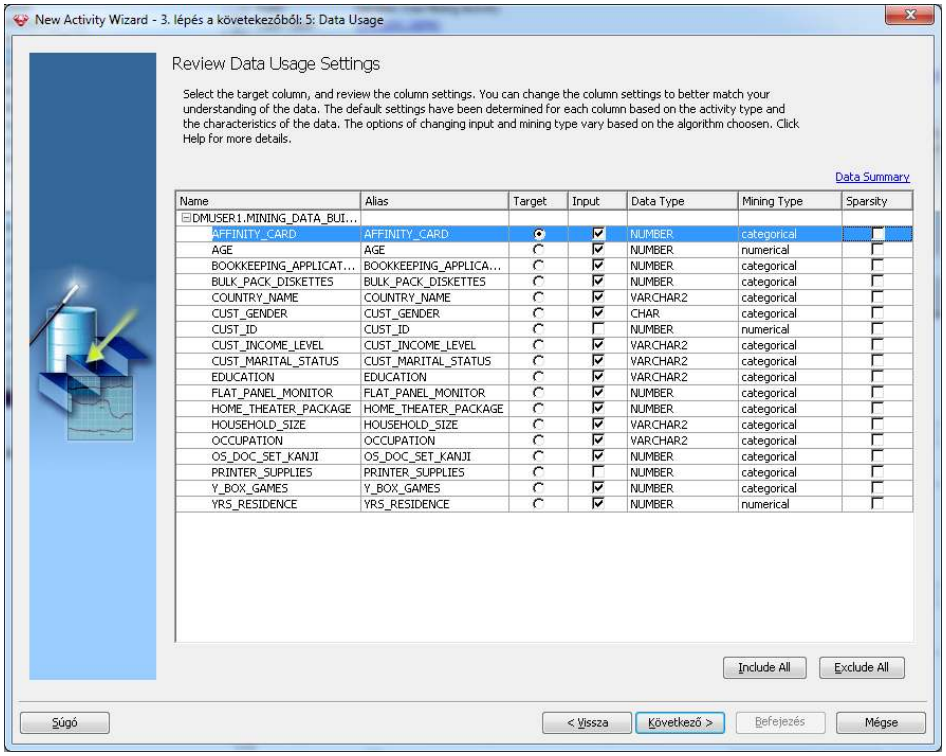
3. Tábla struktúrája.



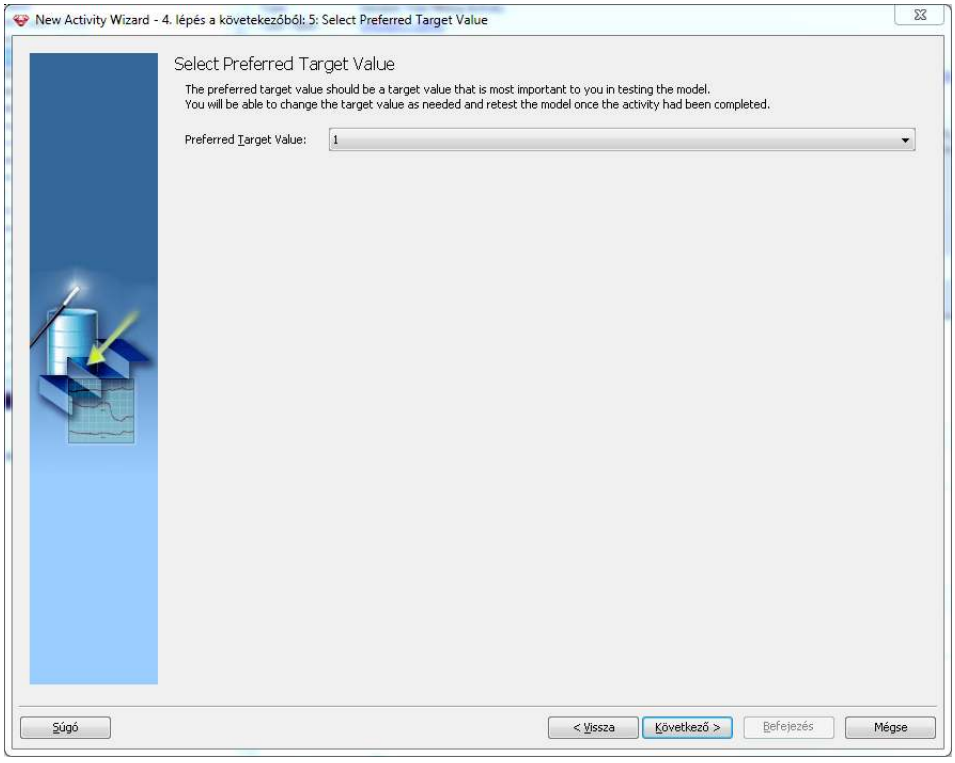
4. Osztályozó algoritmusok.



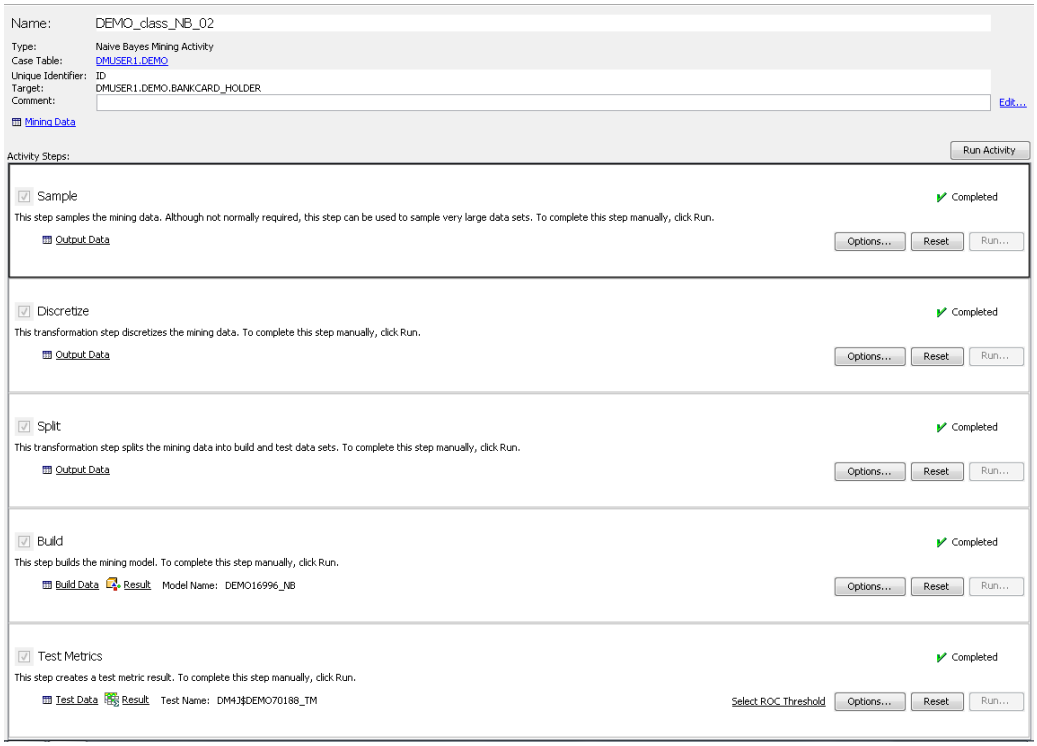
5. Naiv Bayes algoritmus második lépése.



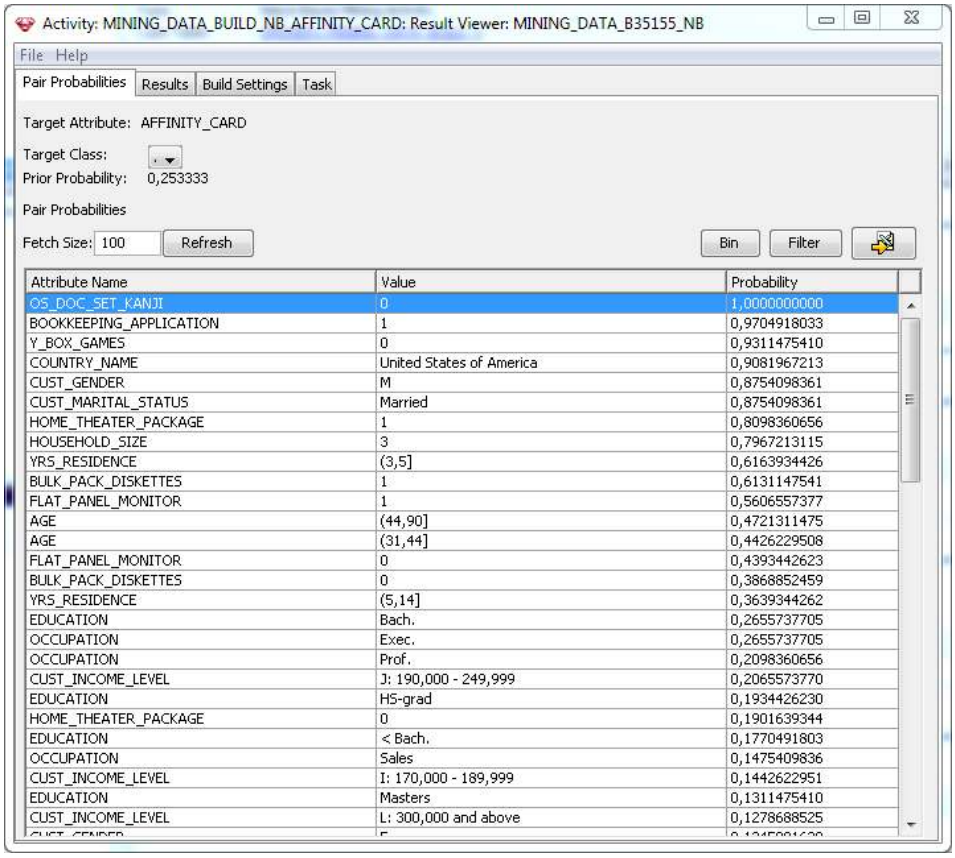
6. Naiv Bayes algoritmus harmadik lépése.



7. Naiv Bayes algoritmus negyedik lépése.

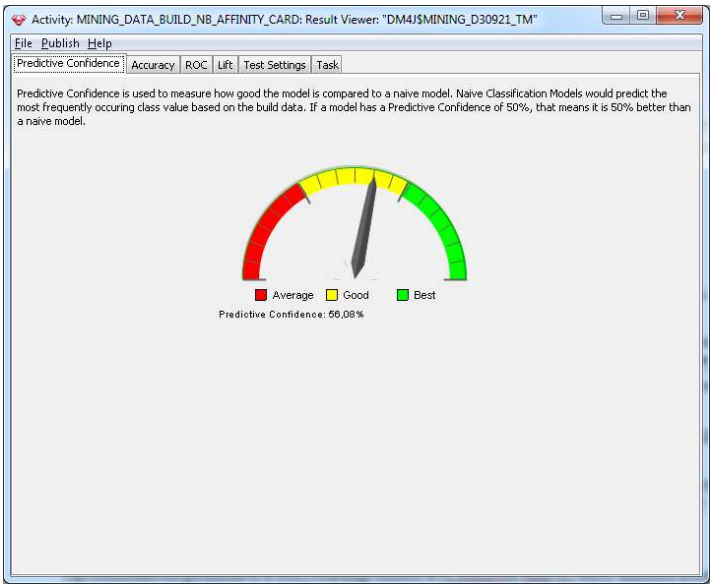


8. Naiv Bayes osztályozó algoritmus futásának lépései.

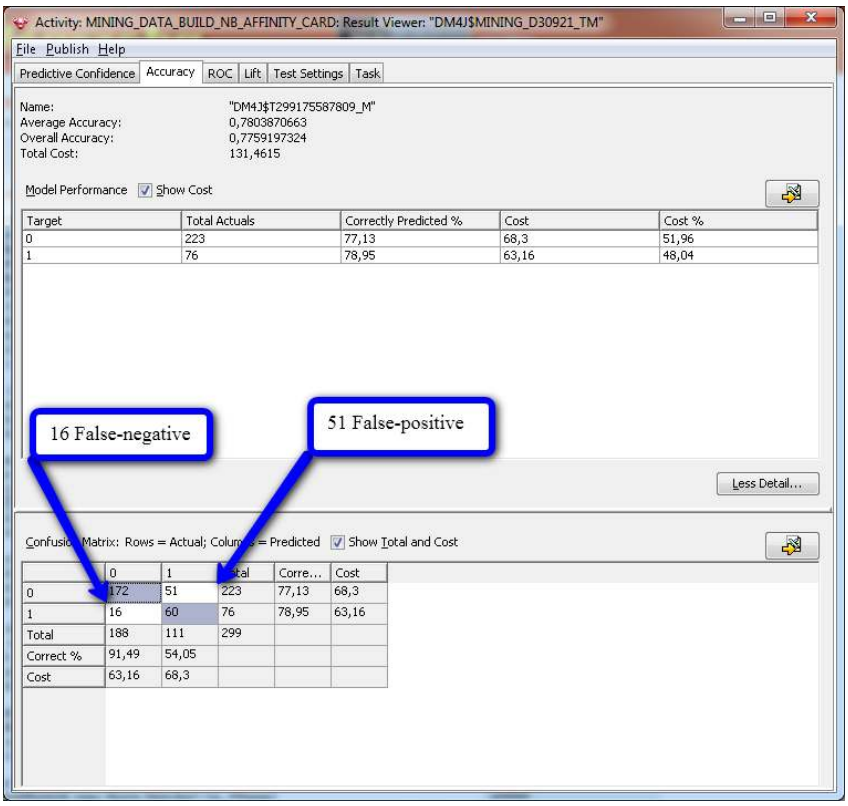


Attribute Name	Value	Probability
OS_DOC_SET_KANJI	0	1,0000000000
BOOKKEEPING_APPLICATION	1	0,9704918033
Y_BOX_GAMES	0	0,9311475410
COUNTRY_NAME	United States of America	0,9081967213
CUST_GENDER	M	0,8754098361
CUST_MARITAL_STATUS	Married	0,8754098361
HOME_THEATER_PACKAGE	1	0,8098360656
HOUSEHOLD_SIZE	3	0,7967213115
YRS_RESIDENCE	(3,5]	0,6163934426
BULK_PACK_DISKETTES	1	0,6131147541
FLAT_PANEL_MONITOR	1	0,5606557377
AGE	(44,90]	0,4721311475
AGE	(31,44]	0,4426229508
FLAT_PANEL_MONITOR	0	0,4393442623
BULK_PACK_DISKETTES	0	0,3868852459
YRS_RESIDENCE	(5,14]	0,3639344262
EDUCATION	Bach.	0,2655737705
OCCUPATION	Exec.	0,2655737705
OCCUPATION	Prof.	0,2098360656
CUST_INCOME_LEVEL	J: 190,000 - 249,999	0,2065573770
EDUCATION	HS-grad	0,1934426230
HOME_THEATER_PACKAGE	0	0,1901639344
EDUCATION	< Bach.	0,1770491803
OCCUPATION	Sales	0,1475409836
CUST_INCOME_LEVEL	I: 170,000 - 189,999	0,1442622951
EDUCATION	Masters	0,1311475410
CUST_INCOME_LEVEL	L: 300,000 and above	0,1278688525

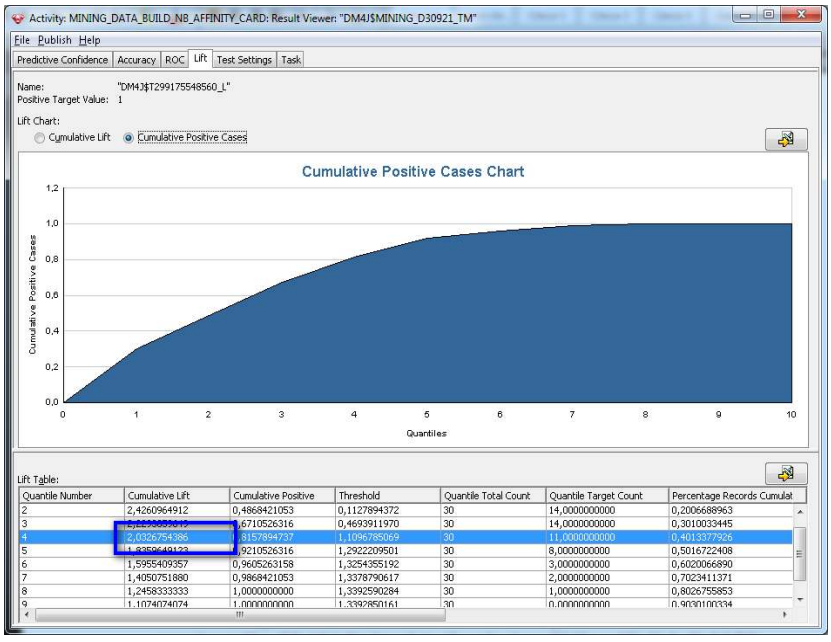
9. A Naiv Bayes osztályozó algoritmus gyakori elempárok előfordulásának valószínűsége.



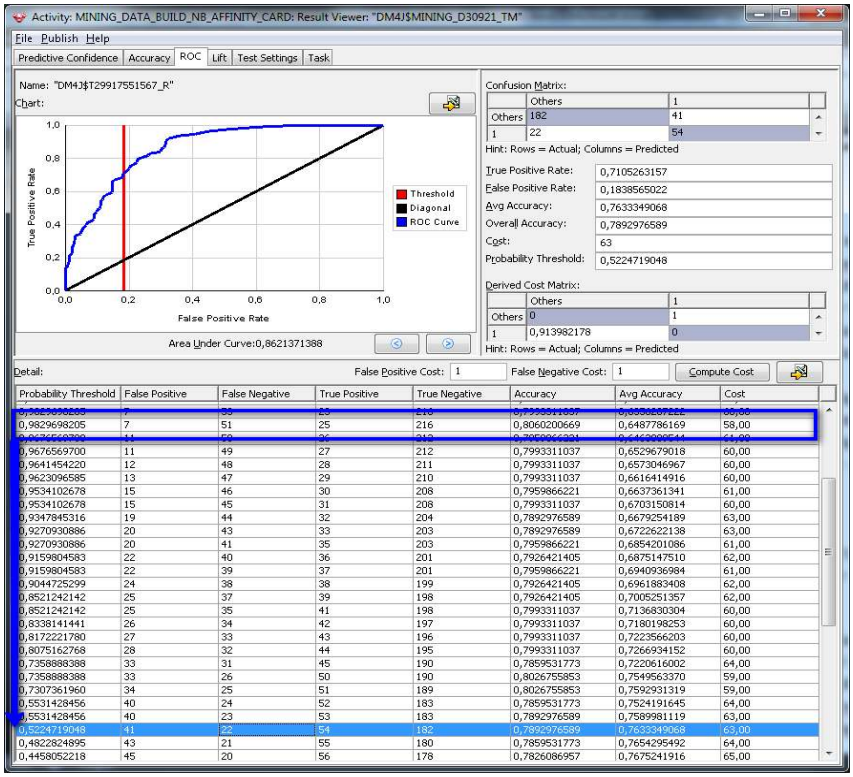
10. Naiv Bayes osztályozó algoritmus által létrehozott modell tesztelésének értékelése.



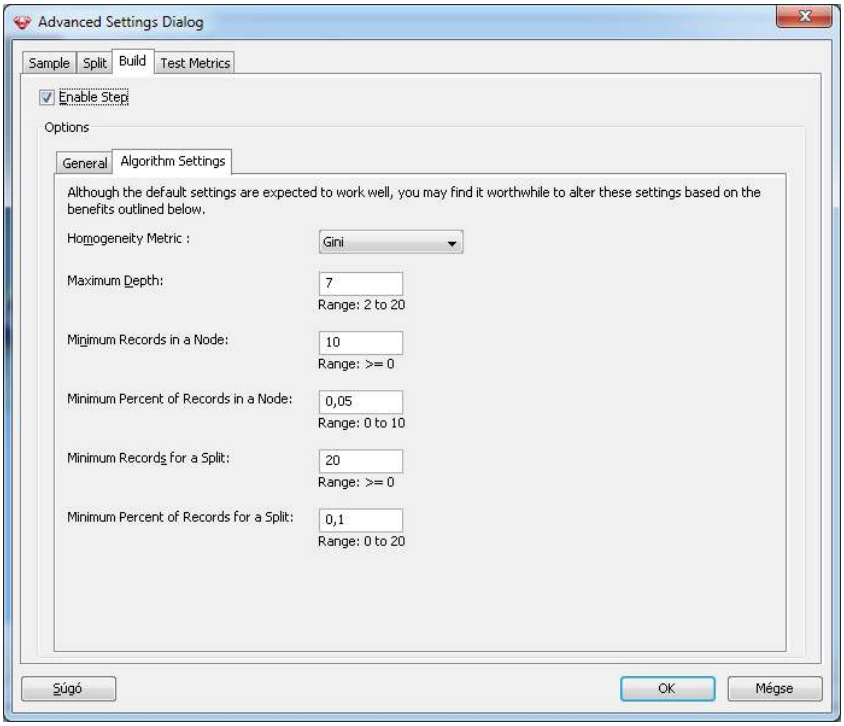
11. Naiv Bayes osztályozó algoritmus pontossága.



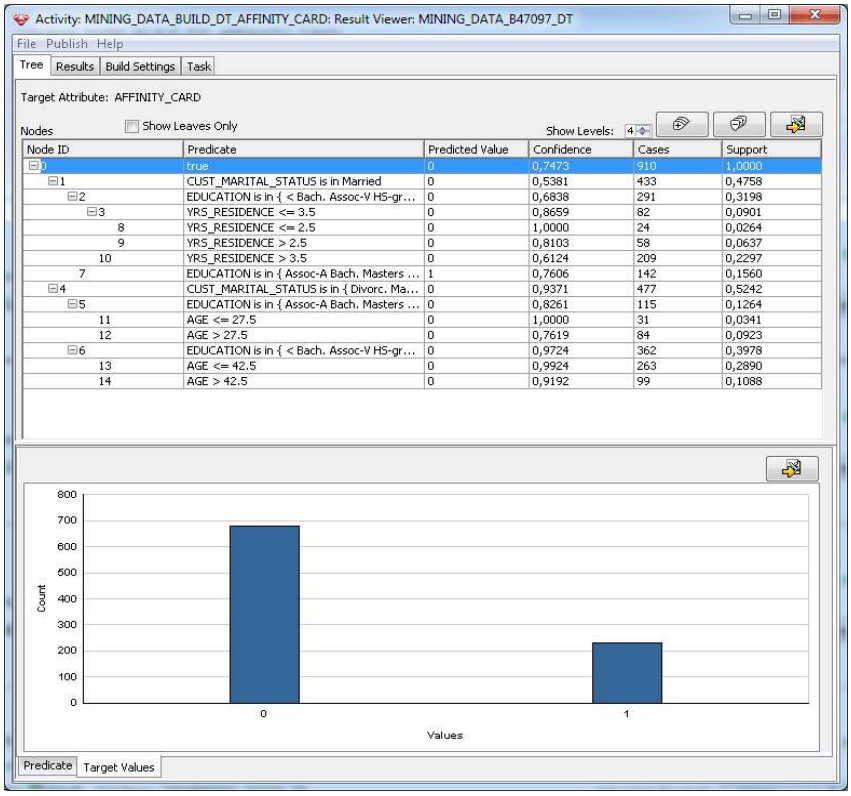
12. Naiv Bayes osztályozó algoritmus nyereség-görbéje, mely a direkt marketingben használatos eredménykimutató eszköz.



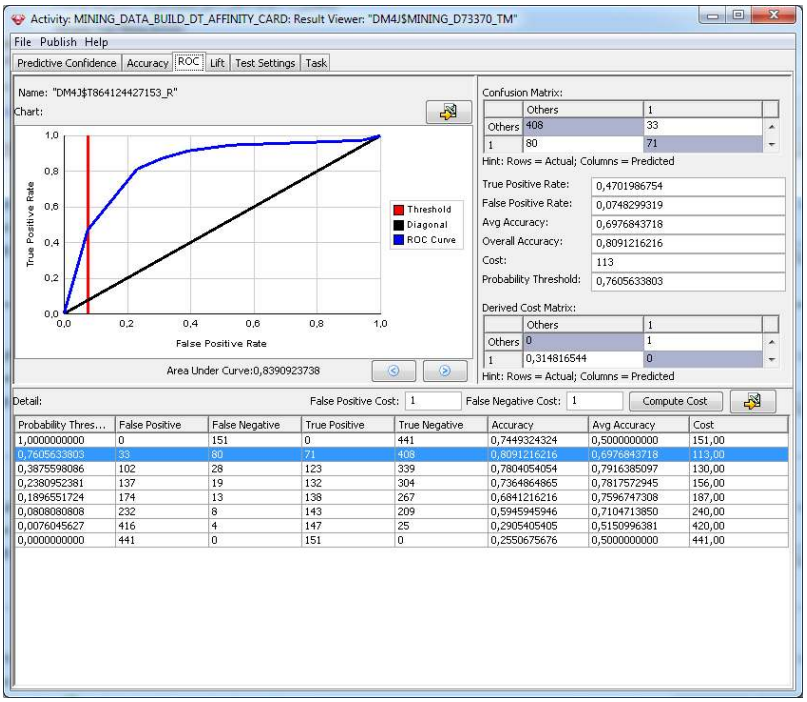
13. Naiv Bayes osztályozó algoritmus ROC analízise.



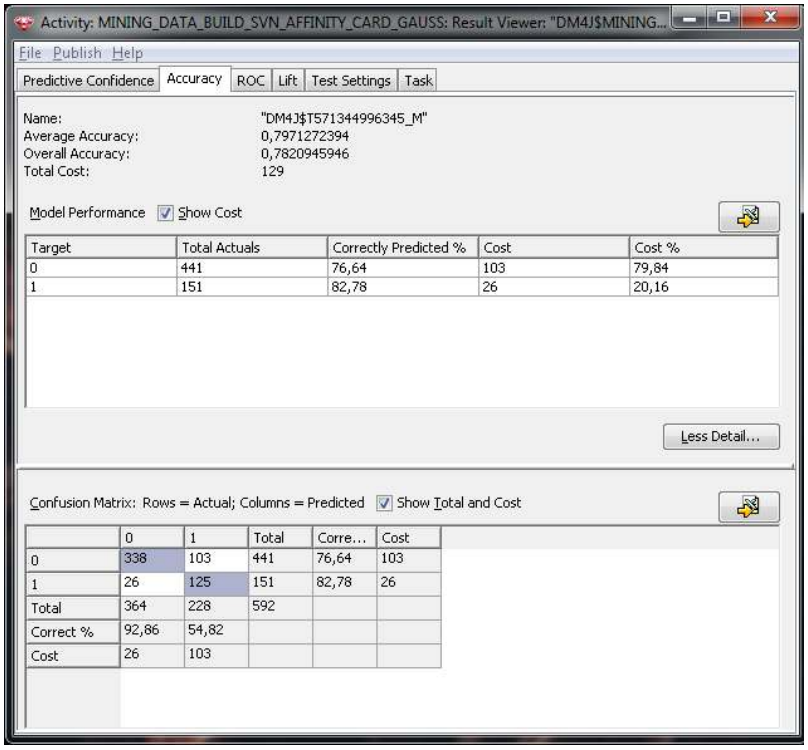
14. Döntési fa algoritmus részletes beállítása.



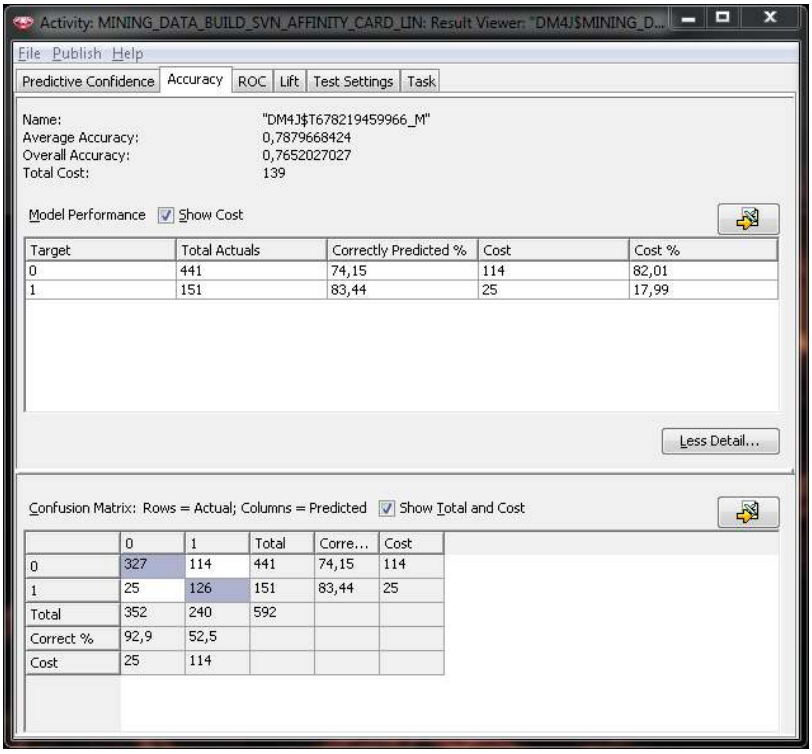
15. Döntési fa algoritmus modelljének áttekintése.



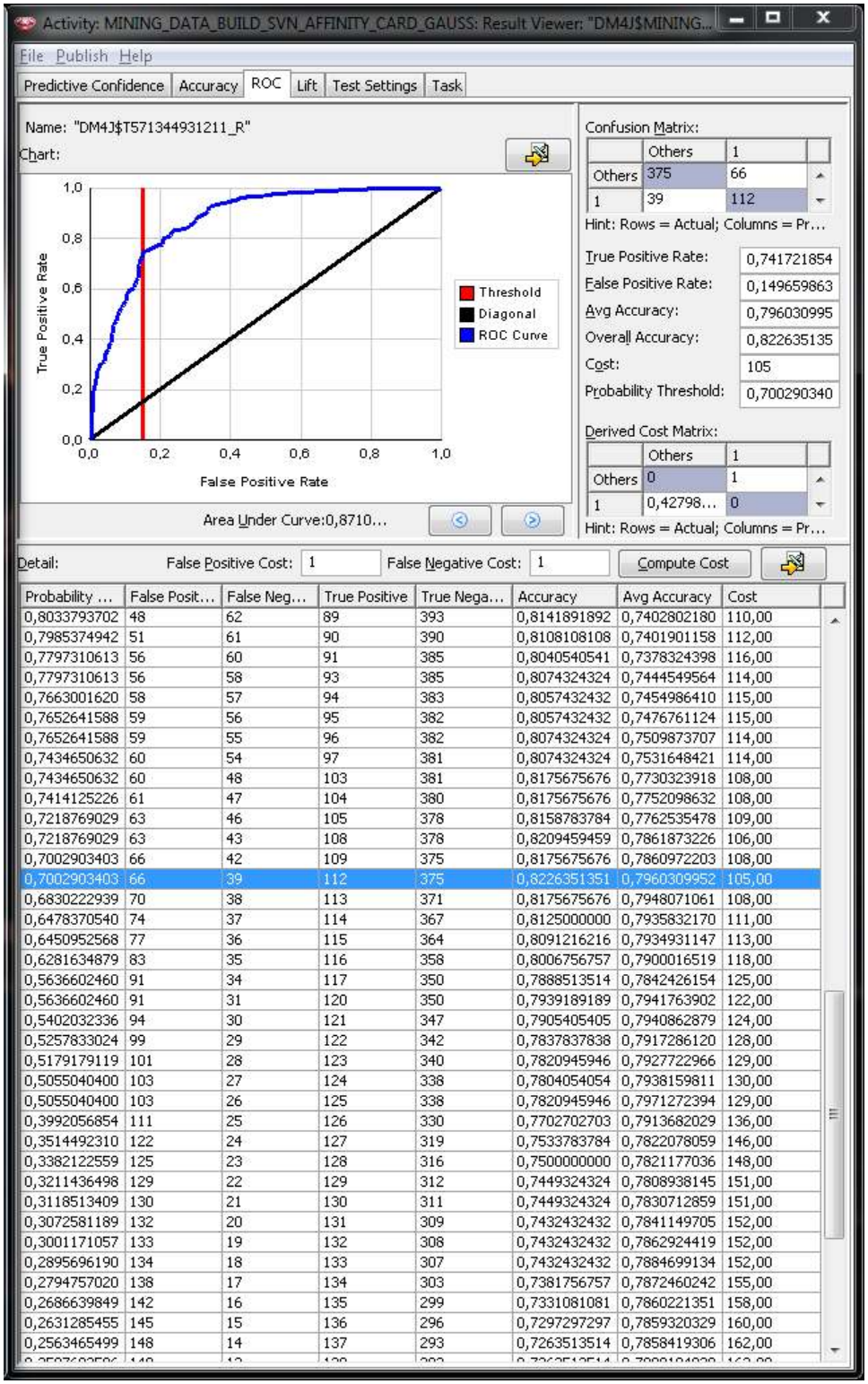
16. Döntési fa algoritmus ROC analízise.



17. SVM algoritmus eredménye Gauss móddal.

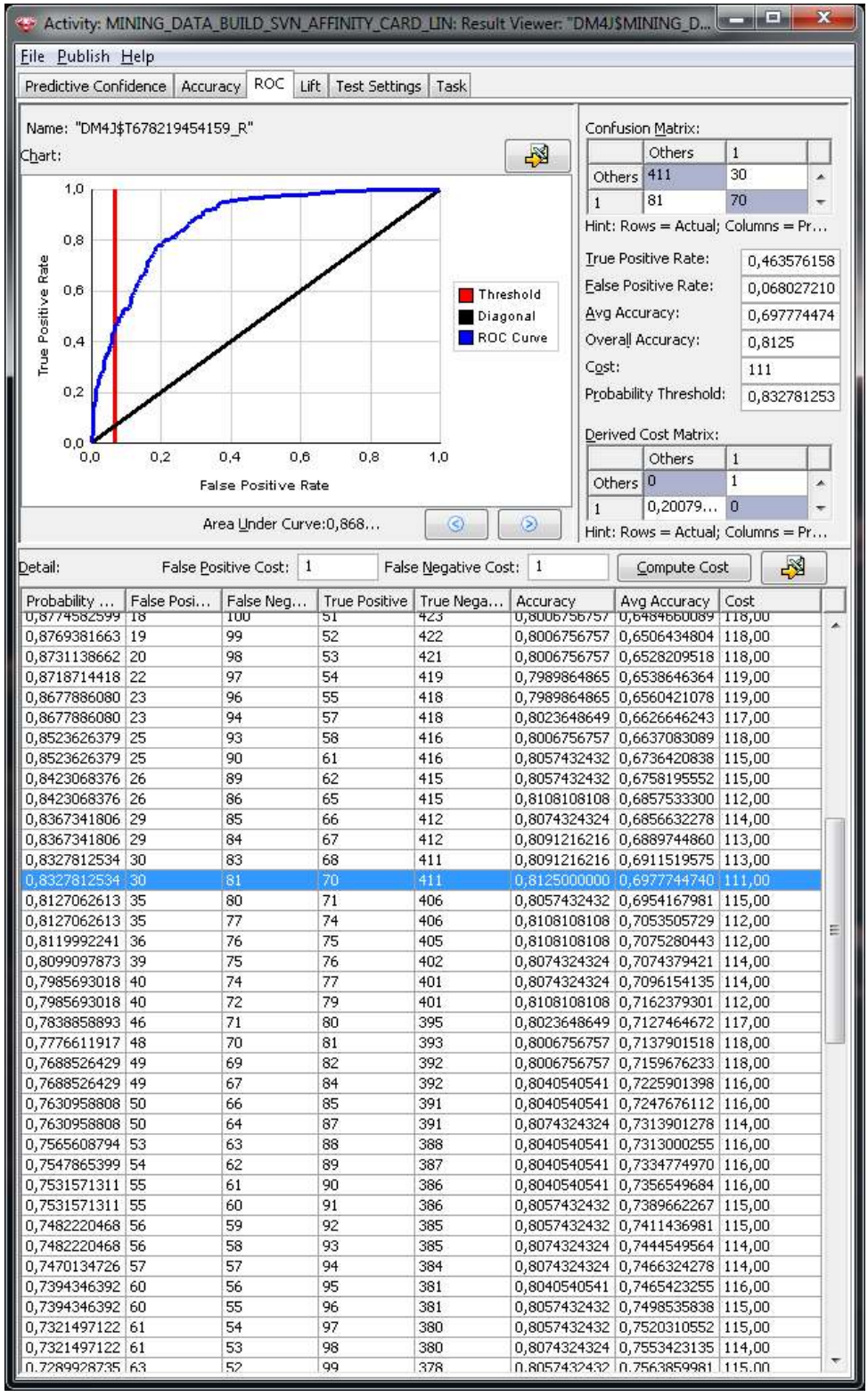


18. SVM algoritmus eredménye Lineáris móddal.



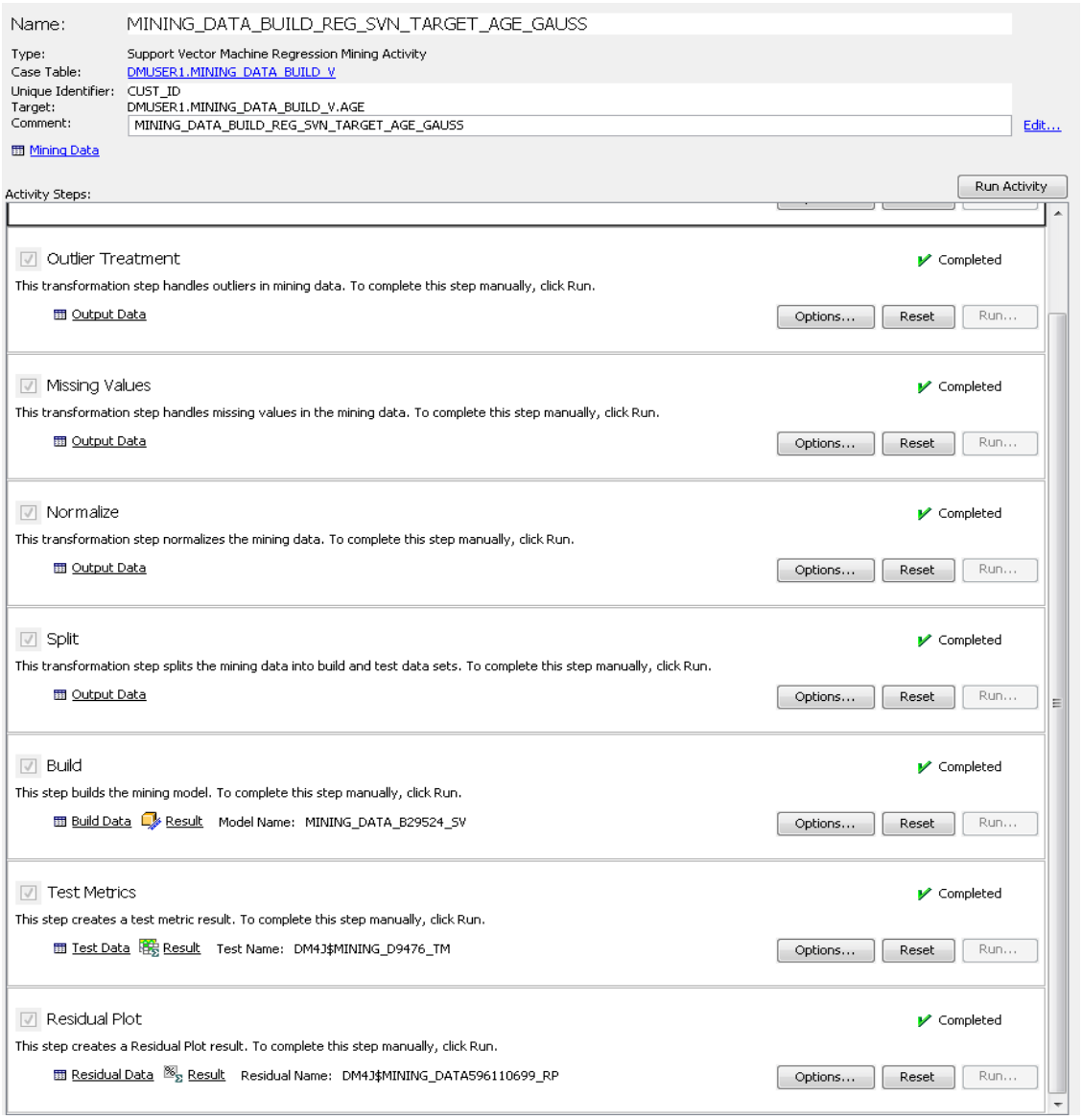
Probability ...	False Posit...	False Neg...	True Positive	True Nega...	Accuracy	Avg Accuracy	Cost
0,8033793702	48	62	89	393	0,8141891892	0,7402802180	110,00
0,7985374942	51	61	90	390	0,8108108108	0,7401901158	112,00
0,7797310613	56	60	91	385	0,8040540541	0,7378324398	116,00
0,7797310613	56	58	93	385	0,8074324324	0,7444549564	114,00
0,7663001620	58	57	94	383	0,8057432432	0,7454986410	115,00
0,7652641588	59	56	95	382	0,8057432432	0,7476761124	115,00
0,7652641588	59	55	96	382	0,8074324324	0,7509873707	114,00
0,7434650632	60	54	97	381	0,8074324324	0,7531648421	114,00
0,7434650632	60	48	103	381	0,8175675676	0,7730323918	108,00
0,7414125226	61	47	104	380	0,8175675676	0,7752098632	108,00
0,7218769029	63	46	105	378	0,8158783784	0,7762535478	109,00
0,7218769029	63	43	108	378	0,8209459459	0,7861873226	106,00
0,7002903403	66	42	109	375	0,8175675676	0,7860972203	108,00
0,7002903403	66	39	112	375	0,8226351351	0,7960309952	105,00
0,6830222939	70	38	113	371	0,8175675676	0,7948071061	108,00
0,6478370540	74	37	114	367	0,8125000000	0,7935832170	111,00
0,6450952568	77	36	115	364	0,8091216216	0,7934931147	113,00
0,6281634879	83	35	116	358	0,8006756757	0,7900016519	118,00
0,5636602460	91	34	117	350	0,7888513514	0,7842426154	125,00
0,5636602460	91	31	120	350	0,7939189189	0,7941763902	122,00
0,5402032336	94	30	121	347	0,7905405405	0,7940862879	124,00
0,5257833024	99	29	122	342	0,7837837838	0,7917286120	128,00
0,5179179119	101	28	123	340	0,7820945946	0,7927722966	129,00
0,5055040400	103	27	124	338	0,7804054054	0,7938159811	130,00
0,5055040400	103	26	125	338	0,7820945946	0,7971272394	129,00
0,3992056854	111	25	126	330	0,7702702703	0,7913682029	136,00
0,3514492310	122	24	127	319	0,7533783784	0,7822078059	146,00
0,3382122559	125	23	128	316	0,7500000000	0,7821177036	148,00
0,3211436498	129	22	129	312	0,7449324324	0,7808938145	151,00
0,3118513409	130	21	130	311	0,7449324324	0,7830712859	151,00
0,3072581189	132	20	131	309	0,7432432432	0,7841149705	152,00
0,3001171057	133	19	132	308	0,7432432432	0,7862924419	152,00
0,2895696190	134	18	133	307	0,7432432432	0,7884699134	152,00
0,2794757020	138	17	134	303	0,7381756757	0,7872460242	155,00
0,2686639849	142	16	135	299	0,7331081081	0,7860221351	158,00
0,2631285455	145	15	136	296	0,7297297297	0,7859320329	160,00
0,2563465499	148	14	137	293	0,7263513514	0,7858419306	162,00

19. SVM algoritmus ROC analízis eredménye Gauss móddal.

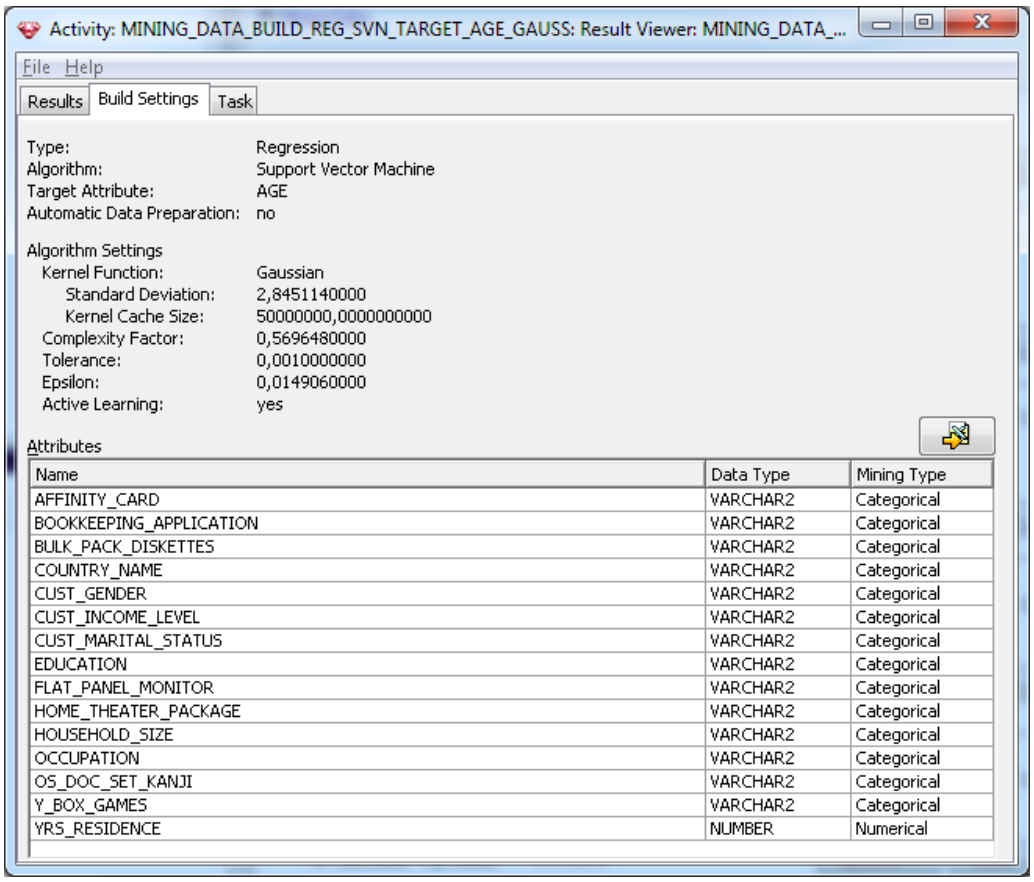


Probability ...	False Posi...	False Neg...	True Positive	True Nega...	Accuracy	Avg Accuracy	Cost
0,8774582599	18	100	51	423	0,8006756757	0,6484660089	118,00
0,8769381663	19	99	52	422	0,8006756757	0,6506434804	118,00
0,8731138662	20	98	53	421	0,8006756757	0,6528209518	118,00
0,8718714418	22	97	54	419	0,7989864865	0,6538646364	119,00
0,8677886080	23	96	55	418	0,7989864865	0,6560421078	119,00
0,8677886080	23	94	57	418	0,8023648649	0,6626646243	117,00
0,8523626379	25	93	58	416	0,8006756757	0,6637083089	118,00
0,8523626379	25	90	61	416	0,8057432432	0,6736420838	115,00
0,8423068376	26	89	62	415	0,8057432432	0,6758195552	115,00
0,8423068376	26	86	65	415	0,8108108108	0,6857533300	112,00
0,8367341806	29	85	66	412	0,8074324324	0,6856632278	114,00
0,8367341806	29	84	67	412	0,8091216216	0,6889744860	113,00
0,8327812534	30	83	68	411	0,8091216216	0,6911519575	113,00
0,8327812534	30	81	70	411	0,8125000000	0,6977744740	111,00
0,8127062613	35	80	71	406	0,8057432432	0,6954167981	115,00
0,8127062613	35	77	74	406	0,8108108108	0,7053505729	112,00
0,8119992241	36	76	75	405	0,8108108108	0,7075280443	112,00
0,8099097873	39	75	76	402	0,8074324324	0,7074379421	114,00
0,7985693018	40	74	77	401	0,8074324324	0,7096154135	114,00
0,7985693018	40	72	79	401	0,8108108108	0,7162379301	112,00
0,7838858893	46	71	80	395	0,8023648649	0,7127464672	117,00
0,7776611917	48	70	81	393	0,8006756757	0,7137901518	118,00
0,7688526429	49	69	82	392	0,8006756757	0,7159676233	118,00
0,7688526429	49	67	84	392	0,8040540541	0,7225901398	116,00
0,7630958808	50	66	85	391	0,8040540541	0,7247676112	116,00
0,7630958808	50	64	87	391	0,8074324324	0,7313901278	114,00
0,7565608794	53	63	88	388	0,8040540541	0,7313000255	116,00
0,7547865399	54	62	89	387	0,8040540541	0,7334774970	116,00
0,7531571311	55	61	90	386	0,8040540541	0,7356549684	116,00
0,7531571311	55	60	91	386	0,8057432432	0,7389662267	115,00
0,7482220468	56	59	92	385	0,8057432432	0,7411436981	115,00
0,7482220468	56	58	93	385	0,8074324324	0,7444549564	114,00
0,7470134726	57	57	94	384	0,8074324324	0,7466324278	114,00
0,7394346392	60	56	95	381	0,8040540541	0,7465423255	116,00
0,7394346392	60	55	96	381	0,8057432432	0,7498535838	115,00
0,7321497122	61	54	97	380	0,8057432432	0,7520310552	115,00
0,7321497122	61	53	98	380	0,8074324324	0,7553423135	114,00
0,7289928735	63	52	99	378	0,8057432432	0,7563859981	115,00

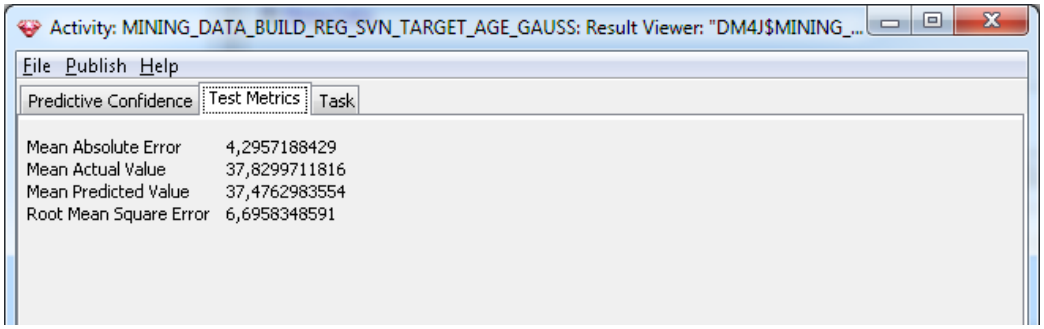
20. SVM algoritmus ROC analízis eredménye Lineáris móddal.



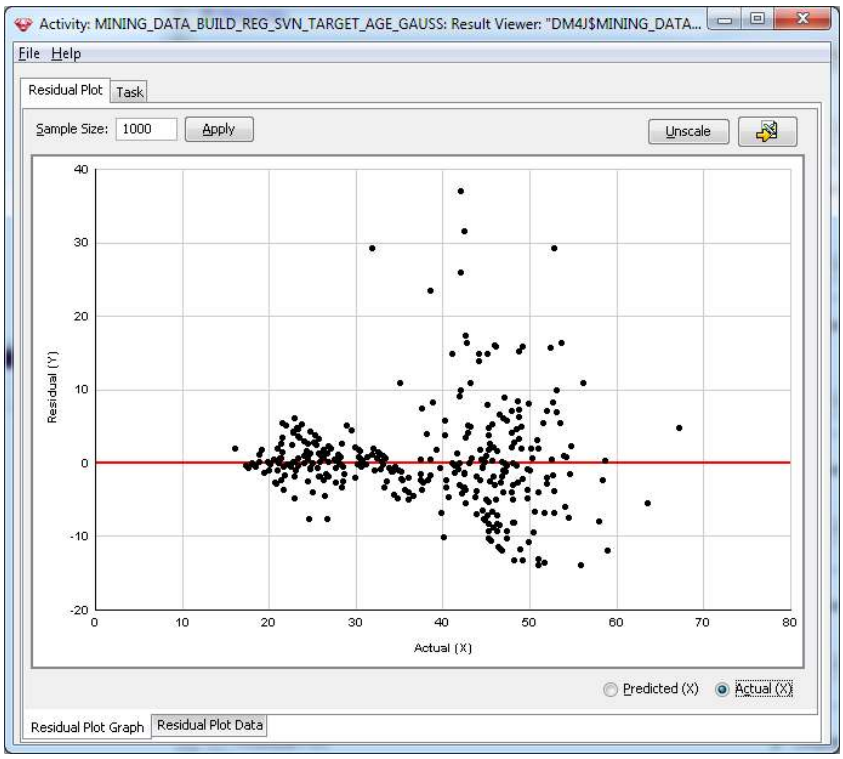
21. SVM algoritmus regressziós esetben való futása.



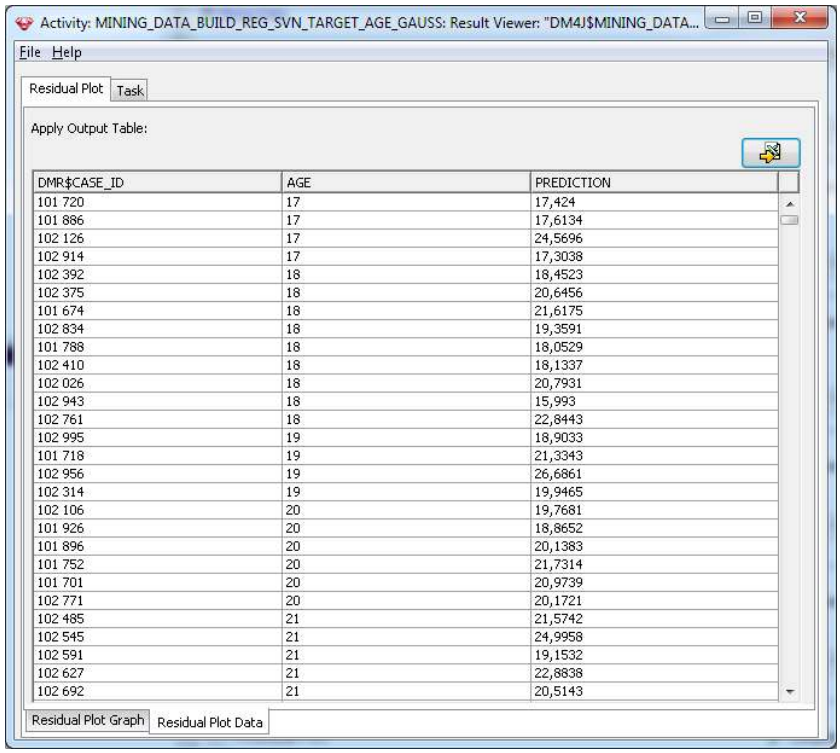
22. SVM algoritmus regressziós modellje.



23. SVM algoritmus regressziós modell tesztelésének eredménye.

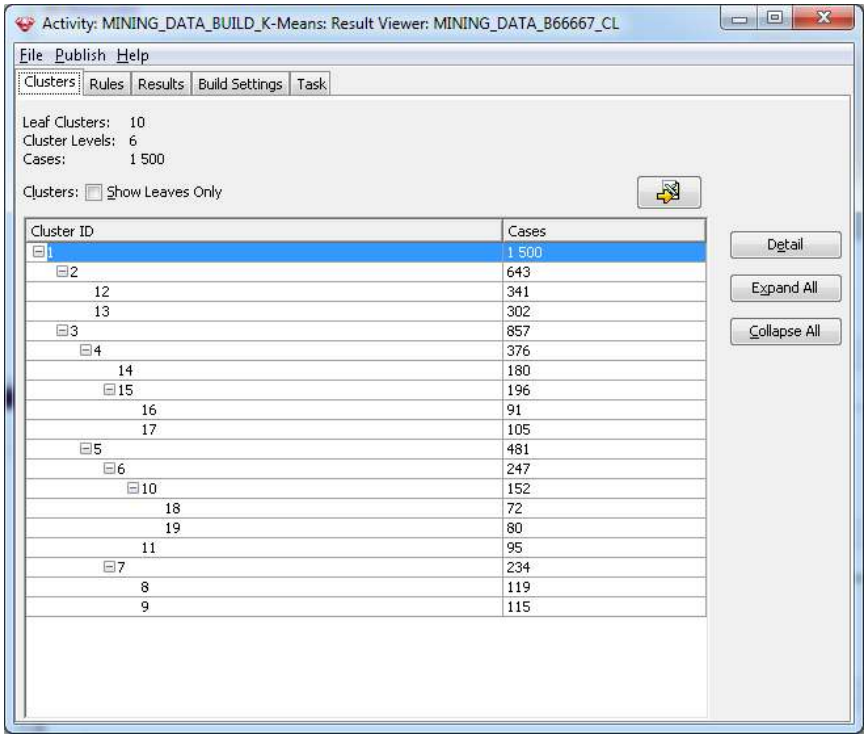


24. SVM algoritmus regressziós modell tesztelésének grafikus eredménye.

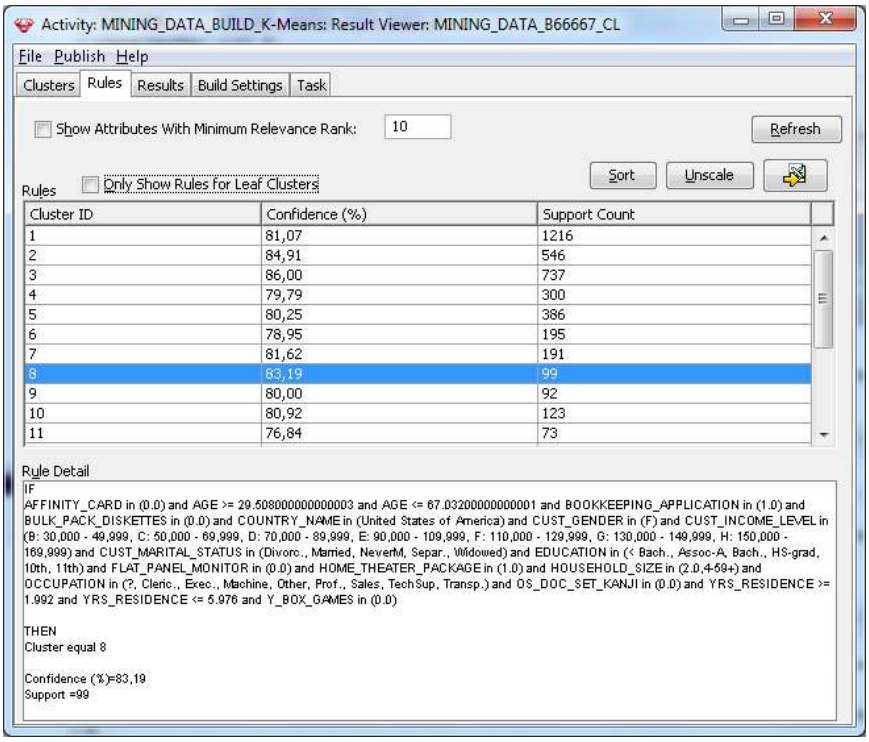


DMR$CASE_ID	AGE	PREDICTION
101 720	17	17,424
101 886	17	17,6134
102 126	17	24,5696
102 914	17	17,3038
102 392	18	18,4523
102 375	18	20,6456
101 674	18	21,6175
102 834	18	19,3591
101 788	18	18,0529
102 410	18	18,1337
102 026	18	20,7931
102 943	18	15,993
102 761	18	22,8443
102 995	19	18,9033
101 718	19	21,3343
102 956	19	26,6861
102 314	19	19,9465
102 106	20	19,7681
101 926	20	18,8652
101 896	20	20,1383
101 752	20	21,7314
101 701	20	20,9739
102 771	20	20,1721
102 485	21	21,5742
102 545	21	24,9958
102 591	21	19,1532
102 627	21	22,8838
102 692	21	20,5143

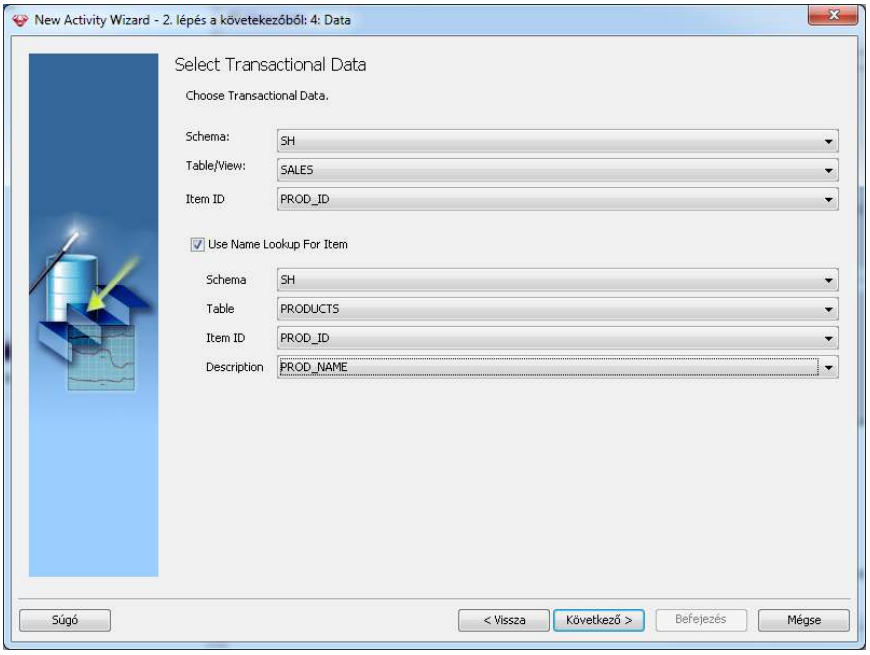
25. SVM algoritmus regressziós modell grafikus eredményének adatai.



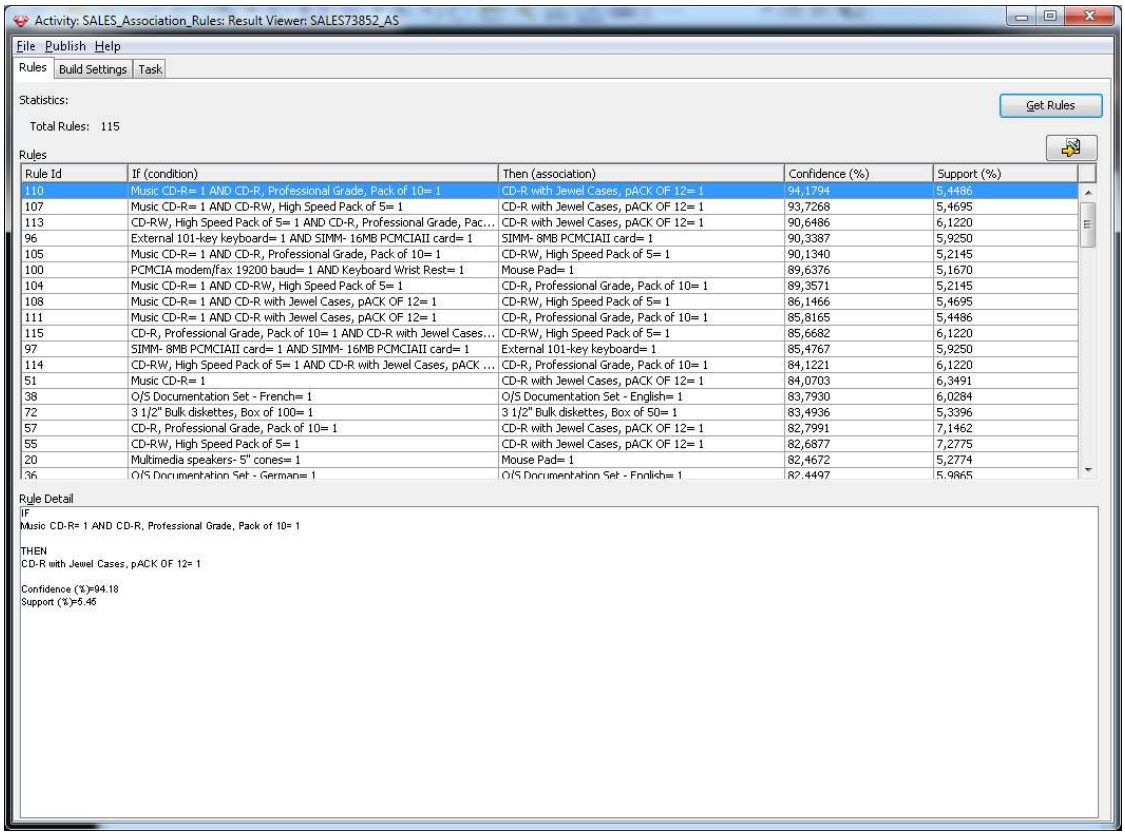
26. K-középpontú algoritmus létrehozott klaszterei.



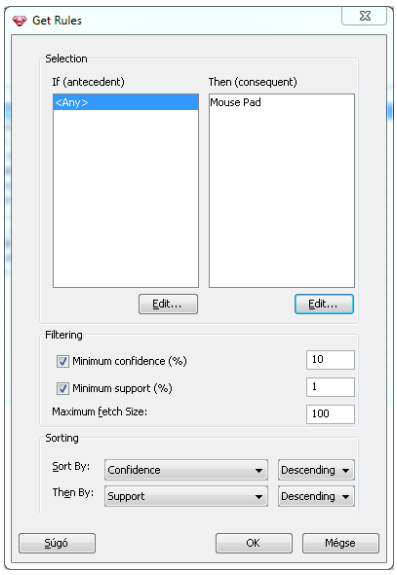
27. Az egyes klaszterekre vonatkozó feltételrendszerek.



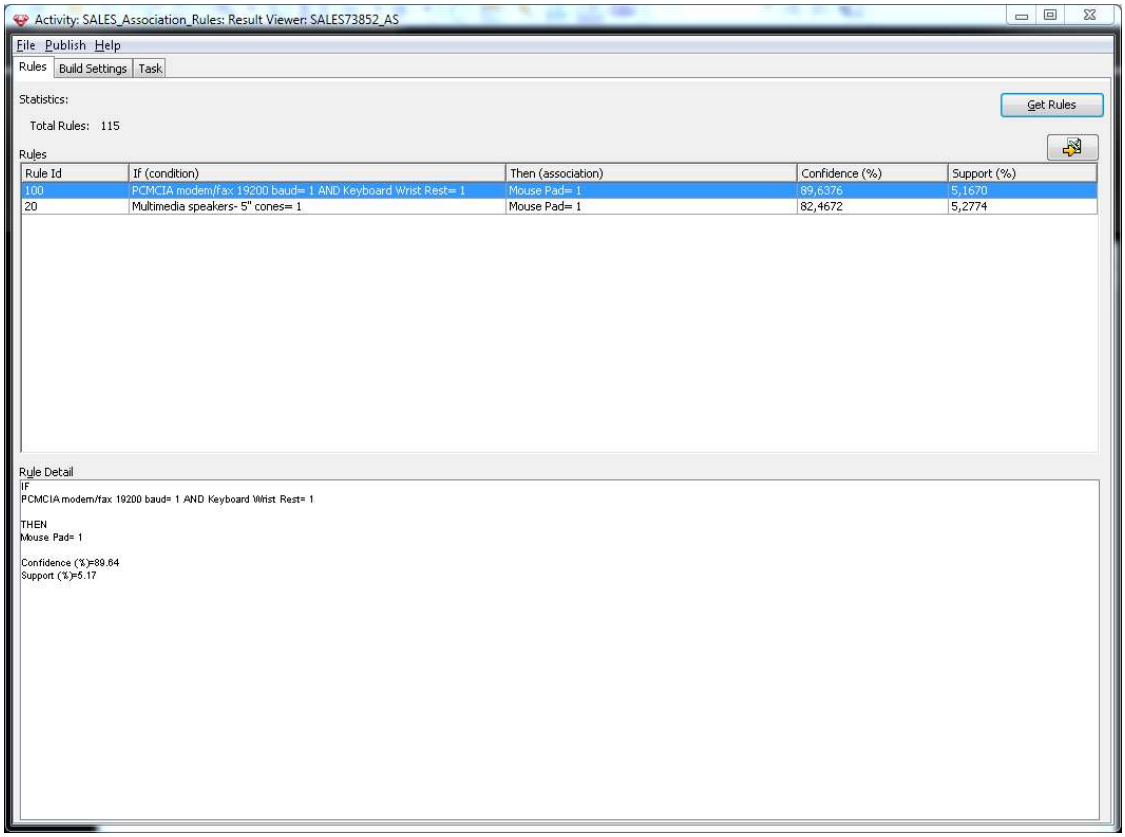
28. Apriori algoritmus esetén a tranzakciókat tartalmazó tábla és a leíró tábla beállítása.



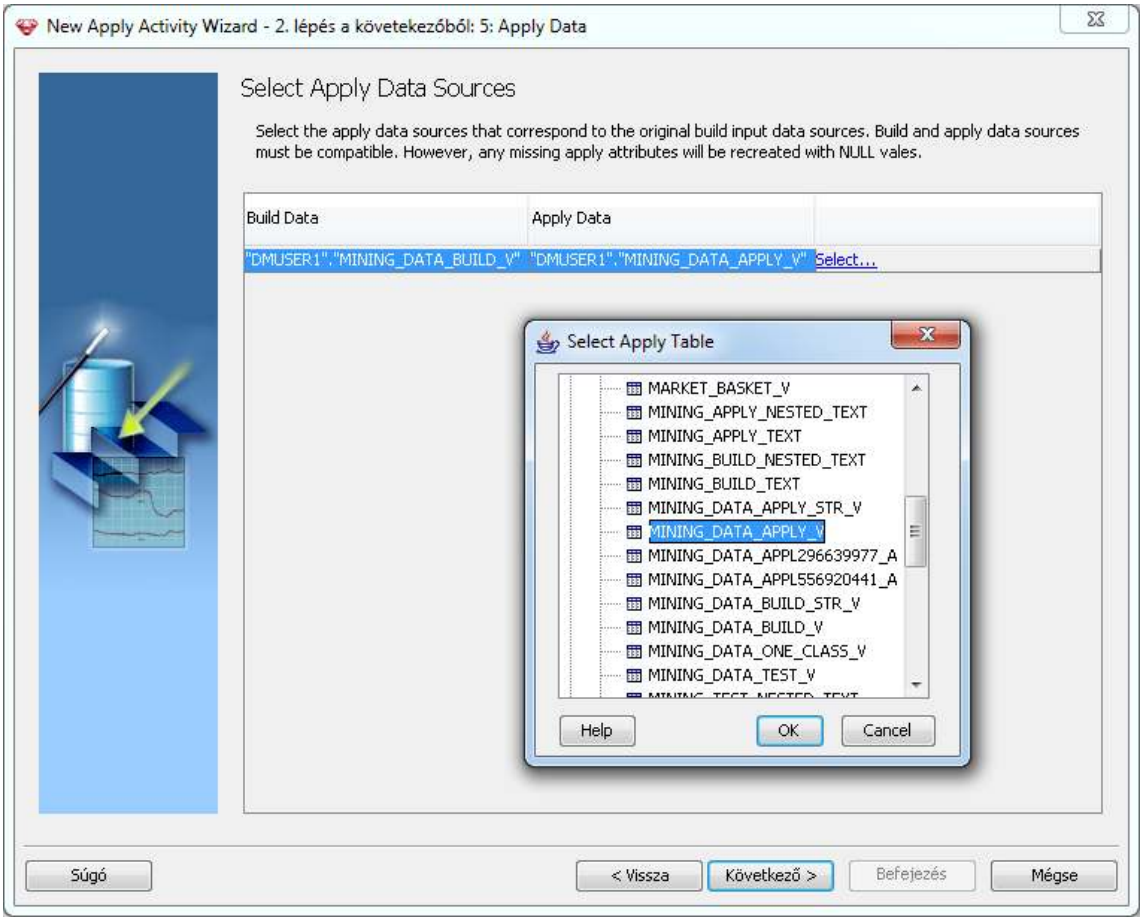
29. Az apriori algoritmus eredménye, gyakori elemhalmazok.



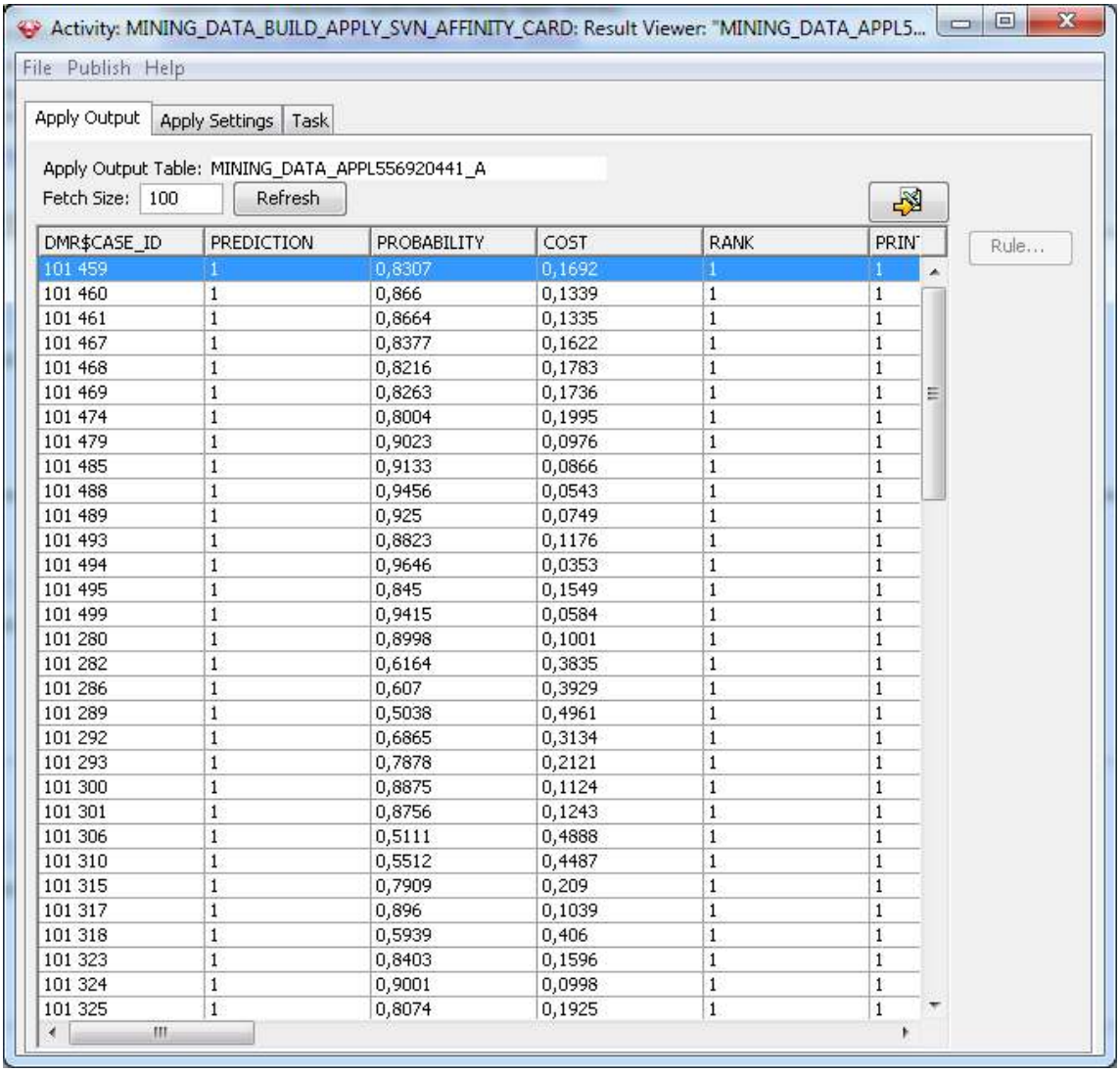
30. A gyakori elemhalmazokra alkalmazható szűrési feltételek.



31. Mely feltételek mellett vásároltak egér alátétet a vásárlók.



32. Választunk egy adattáblát melyen az elkészült modellt alkalmazzuk.



DMR$CASE_ID	PREDICTION	PROBABILITY	COST	RANK	PRIN
101 459	1	0,8307	0,1692	1	1
101 460	1	0,866	0,1339	1	1
101 461	1	0,8664	0,1335	1	1
101 467	1	0,8377	0,1622	1	1
101 468	1	0,8216	0,1783	1	1
101 469	1	0,8263	0,1736	1	1
101 474	1	0,8004	0,1995	1	1
101 479	1	0,9023	0,0976	1	1
101 485	1	0,9133	0,0866	1	1
101 488	1	0,9456	0,0543	1	1
101 489	1	0,925	0,0749	1	1
101 493	1	0,8823	0,1176	1	1
101 494	1	0,9646	0,0353	1	1
101 495	1	0,845	0,1549	1	1
101 499	1	0,9415	0,0584	1	1
101 280	1	0,8998	0,1001	1	1
101 282	1	0,6164	0,3835	1	1
101 286	1	0,607	0,3929	1	1
101 289	1	0,5038	0,4961	1	1
101 292	1	0,6865	0,3134	1	1
101 293	1	0,7878	0,2121	1	1
101 300	1	0,8875	0,1124	1	1
101 301	1	0,8756	0,1243	1	1
101 306	1	0,5111	0,4888	1	1
101 310	1	0,5512	0,4487	1	1
101 315	1	0,7909	0,209	1	1
101 317	1	0,896	0,1039	1	1
101 318	1	0,5939	0,406	1	1
101 323	1	0,8403	0,1596	1	1
101 324	1	0,9001	0,0998	1	1
101 325	1	0,8074	0,1925	1	1

33. A megbízható ügyfelek listája melyet az SVM algoritmussal készített modellel nyertünk.