Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Döntéstámogatási módszerek alkalmazása teljesítménymodellek adatvezérelt szintézisében

TDK-dolgozat

Készítette:

Burján Dezső

Konzulens:

Gönczy László

2017

Tartalomjegyzék

Kivonat

Napjainkban a rendelkezésre álló adatmennyiség mellett kihívás egy megalapozott döntés meghozása, hiszen sokszor számos lehetőség közül tudunk választani és a befolyásoló tényezők száma is igen széles skálán mozog. Az üzleti életből származó problémák mellett hasonló kérdések merülnek fel informatikai rendszerek működésének megértése és befolyásolása esetén.

Előfordulhat továbbá, hogy a rendszerekben ismert változók közötti összefüggésekről nem teljes az ismeretünk, a konfiguráció vagy a rendszer terhelése az idő során változnak.

A manapság ismert monitorozási módszerek végrehajtása után rengeteg nyers adathoz sikerül hozzáférnünk. Ezeket az adatokat legelőször meg kell értenünk, majd feldolgoznunk és elemeznünk kell őket, csak azok után lehet a segítségükkel különböző döntéseket ajánlani a felhasználók számára.

A megkapott információk sokszor nagyon komplexek és az emberi szem számára átláthatatlanok. Ilyenkor hasznosak a modern, gyakorlatban is használt adatelemzési és döntéstámogatási módszerek, melyek segítségével a számítógép segít nekünk a döntés meghozásában, ráadásul ez a folyamat automatizálható is.

Ezek a döntések segíthetnek abban, hogy azonosítani tudjuk a fontosabb fejlesztéseket és monitorozási pontokat is. Az is fontos, hogy olyan modelleket származtassunk a rendszerekből, amelyek hordozhatóak, így nem csak egy konkrét rendszer konfigurációra érvényesek, hanem a rugalmasságuknak köszönhetően akár konfiguráció változás esetén is.

Munkám során bemutatok egy módszert, amely a modern döntéstámogatás és adatelemezés segítségével segít a mérnöki gyakorlatban felbukkanó döntések meghozásában. Egy saját teljesítménymodellt is létrehozok az egyik informatikai rendszer terheléstesztje után nyert adatok segítségével. Dolgozatomban azt is bemutatom egy esettanulmányon keresztül, hogy ezeket a módszereket a gyakorlatban hogyan lehet használni. A módszerem segítségével azt is megtudhatjuk, hogy mely változók fontosabbak egy informatikai rendszerben. A hatékonyságukat egy esettanulmányon fogom bemutatni, valamint megmutatom, hogy eredményeiket hogyan lehet felhasználni teljesítménymodellek szintézise során.

Abstract

Although the amount of available data is growing, decision making remains a challenging task due to the number of factors and their interdepencies. While such problems are well-known in business life we can find similar questions while we try to understand and control the IT systems. Moreover the previous problem could be harder, because often the configuration or the system workload characteristics may vary over time.

Modern monitoring methods typically gather a huge amount of raw data. The information behind this data can be complex and hard to interpret for an operator. Modern methods of data analysis and decision support used in this work can help in decision making, while this process can partly be automated as well.

Such decisions can help in identification of necessary system improvements or determine monitoring points.

The models created for complex systems are also an important as these should be portable so that reuse in other configurations can be supported.

In this work I present a method to apply modern technologies of decision support and data analysis in engineering decisions. The input data is gathered from the simulations and the performance measurements of the systems. I also show how we can use the previously mentioned methods for making the models.

I show how we can use the decision support methods while taking account the informations gathered from the system models and expert knowledge. I also evaluate these methods and analyse their effectiveness on a real case study. I also show how the results of such methods can be used in synthesizing performance models.

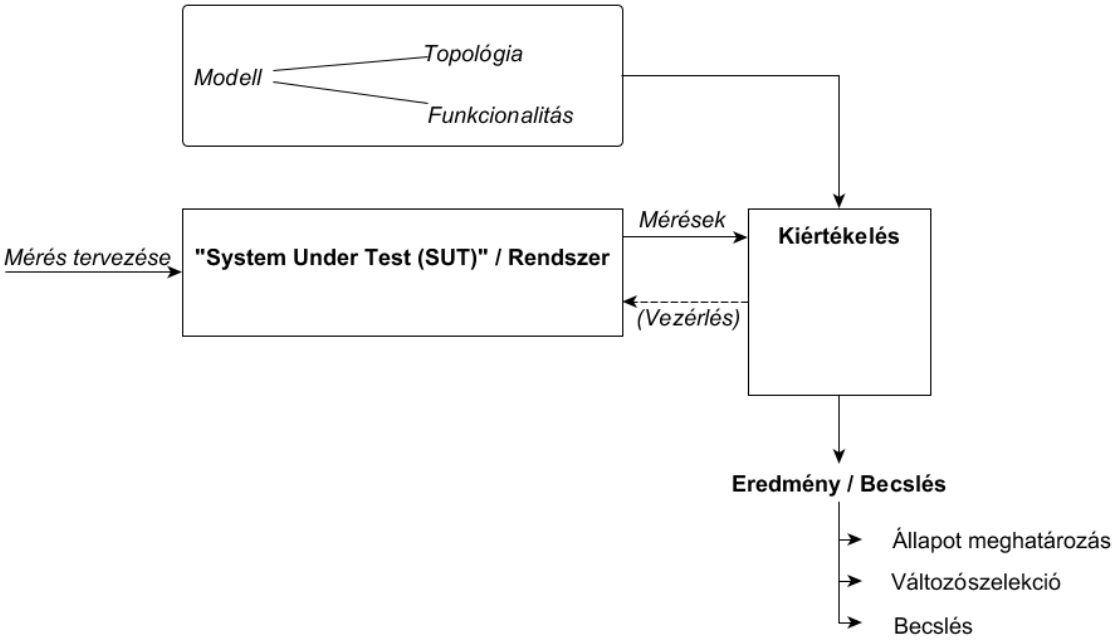
1. fejezet

Bevezetés

Az IT rendszerek növekvő komplexitása és a felhő technológiák is váratlan hatással lehetnek a rájuk épülő szolgáltatásokra. Napjainkra jellemző, hogy az infrastruktúra fenntartója sok adatot gyűjt, de nem használja fel azokat, hanem például ha valamilyen hiba van, akkor utólag nézik meg, hogy mi is lehetett a kiváltó ok.

Dolgozatomban egy olyan módszert szeretnék bemutatni, amely segítségével megtudhatjuk, hogy mely adatokat érdemes gyűjtenünk, illetve mérnünk, segít meghatározni egy komplex rendszer állapotát, és felderíteni annak lényeges szűk keresztmetszeteit, előrejelző képességének segítségével elkerülhetjük a hibás / nem kívánt állapotokat. Az általa kapott eredményekből rendszer felügyeleti szabályokat is paraméterezhetünk.

A munkám során informatikai rendszerek teljesítménymodelljének készítésével és azok elemzésével foglalkoztam. Egy konkrét, felhőalapú rendszer mért adataiból kiindulva adatelemzési módszereket hajtottam végre és így készült el a teljesítménymodellem. Több tanulmány[21], [22], [12], [18] is született már a témával kapcsolatban, azok is hasonló problémákra keresik a választ, különböző statisztikai elemzések segítségével. Munkám során a gyakorlatban is használt adatelemzési módszereket használtam, amelyek jobbak a statisztikai elemzésnél. Ráadásul a módszerem jobban kezeli azt, ha több mérés adatait kell egyszerre vizsgálni.



1.1. ábra. A dolgozatom követett megközelítés.

Az 1.1-es ábrán a tesztelendő rendszer az a felhő alapú szolgáltatást nyújtó IMS rendszer (Clearwater), amin a MIT tanszék méréseket hajtott végre. A modell a mért adatok formája és jellege, melyek a mérés kötöttségével állnak kapcsolatban. A tervezésnek az általam használt módszerek és programok felelnek meg. A dolgozatom kulcstémája a mérési adatok kiértékelése, amelyeket később a vezérléshez is fel lehet használni. A bevezetés után az 2. fejezetben röviden bemutatom magát a teljesítménymodellt és annak célját, majd a módszereket és algoritmusokat, amelyeket munkám során használtam. Ezután a rendszert és az adatokat fogom ismertetni, illetve tanulmányokat, melyeket korábban már mások elvégeztek a témával kapcsolatban. Az olvasó azt is megtudhatja majd, hogy a jelen munkámban a teljesítménymodell változói hogyan felelnek meg a gyakorlatban általam elemzett rendszer bemeneteinek, kimeneteinek és állapotainak is. A 3. fejezetben bemutatom a módszerem eredményét az adathalmazon. 4. fejezetben kiértékelem a módszerem eredményeit. Az 5. fejezetben pedig röviden összefoglalom a munkámat.

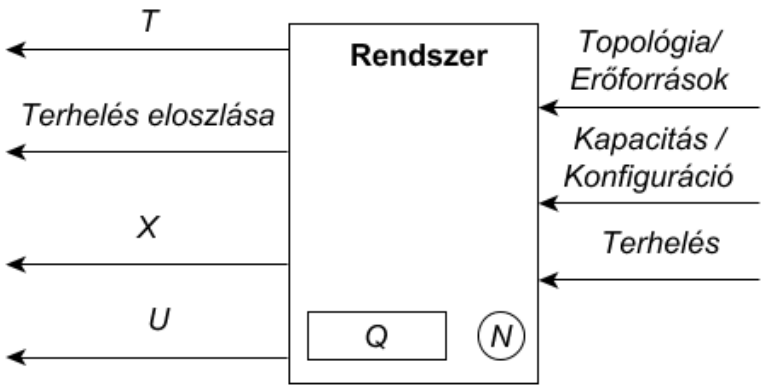
2. fejezet

Teljesítménymodellek és döntéstámogatás

A fejezetben ismertetem a teljesítménymodell definícióját. Ezután leírom, hogy milyen döntéstámogatási, adatelemezési módszereket és algoritmusokat használtam. A fejezet végén pedig az esettanulmányt fogom bemutatni, írok majd az adatokról és magáról a rendszerről, amelyből azok származnak. A fejezet végén pedig bemutatom az adatokon alapuló, általam a Clearwater rendszer leírására létrehozott teljesítménymodellt. A statisztika, és a döntéstámogatás azért volt lényeges, mert így a tényleges, mért adatok alapján tudtam elkészíteni a magas szintű teljesítménymodellemet.

2.1. Teljesítménymodell bemutatása

A teljesítménymodellben van egy rendszer, melynek bemenetei (erőforrások, beállítások) lehetnek ismertek vagy ismeretlenek, kimenetei és belső állapotváltozói is vannak, melyeket a gyakorlati életben sokszor szintén nem ismerünk. A feladatom az volt, hogy különböző adatelemzési és döntéstámogatási módszerek segítségével előrejelzéseket, osztályzásokat tudjak tenni a rendszer állapotára.



2.1. ábra. A teljesítménymodell felépítése.

A rendszerünk bemenetei:

- **Erőforrások / Topológia**
  Az első bemenet adja meg, hogy milyen erőforrások állnak rendelkezésre a rendszerünk számára, a második pedig azt, hogy mi a kapcsolat közöttük.

- **Kapacitás / Konfiguráció**
  Előbbi a rendszerünk kapacitását adja meg, amely összegzi az erőforrásokat, vagyis annak egy tulajdonsága. Az utóbbi segítségével pedig a belső állapotokat szabályozhatjuk, pl. a várakozási sor maximális mérete.

- **Terhelés**
  Ezzel a bemenettel tudjuk konfigurálni a rendszerünk aktuális terhelését, ezt a gyakorlatban nem mindig ismerjük.

A rendszerünk kimenetei:

- **T (Válaszidő)**
  Ez a metrika adja meg, hogy a rendszerünk mennyi idő alatt válaszolt az aktuális bemenetekre.

- **Terhelés eloszlása**
  Ez adja meg, hogy a rendszerünk bemeneteként kapott terhelésnek mi az eloszlása.

- **X (Átbocsátás)**
  Az átbocsátás adja meg, hogy mi a rendszerünk megvalósítható reális teljesítménye az általunk vizsgált időszakban.

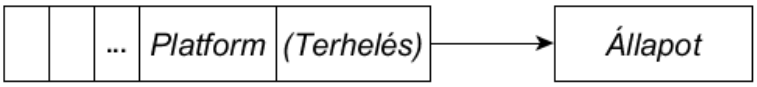
- **U**
  A kihasználtság adja meg, hogy a globális teljesítménykorlátoktól nagyjából mennyire távol működik a rendszerünk.

A rendszerben megtalálható belső változók:

- **Q**
  A várakozási sor jelenlegi mérete.

- **N**
  A rendszerünkben tartózkodó kérések átlagos száma.

A dolgozatomban a prediktív analízis segítségével próbálok meg választ adni arra a kérdésre, hogy a rendszer bizonyos bemeneti értékei és állapotváltozói mellett mi lesz az állapota és eközben arra kérdésre is választ adok, hogy melyik bemenetek és állapotváltozók lesznek a legfontosabbak. Az ehhez felhasznált módszereket és algoritmusokat a következő szekcióban fogom bemutatni.



2.2. ábra. A dolgozatom céljának összefoglalása.

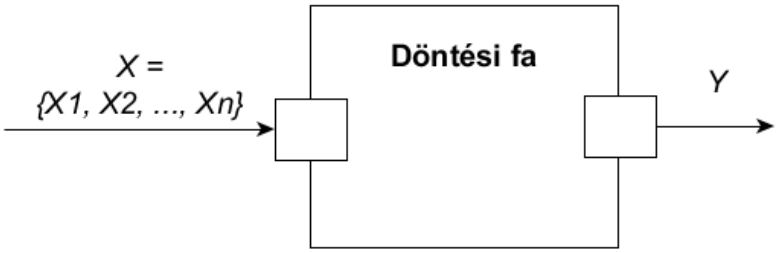
2.2. Predikciós módszerek

Ebben a szekcióban bemutatom a gyakorlatban előszeretettel használt két gépi tanulási módszert, amely segítségével előrejelzést és klasszifikációt tudunk végrehajtani a prediktor változók alapján a célváltozóra. Leírom ezek előnyeit és hátrányait, valamint bemutatom azokat a metrikákat, amelyek segítségével megtudhatjuk, hogy mennyire lettek pontosak az adatokhoz illesztett modellek.

2.2.1. Döntési fa

A *döntési fa* egy döntéstámogató eszköz, amely egy fa gráfon jeleníti meg a döntéseinket és azok kimeneteleit. Az ábrán rajta lehetnek akár az egyes események valószínűsége, költsége, hatékonysága. Az operációkutatás egy fontos eszköze, de számos más tudományág is előszeretettel használja. Ha egy fekete dobozként tekintünk rá, akkor a bemenete egy attribútumhalmaz ($X = \{X_1, X_2, ..., X_n\}$), melyben a változók értékei lehetnek numerikus vagy kategorikus adatok. A kimenetének (Y) típusa vagy egy osztály (Klasszifikációs fa), vagy egy numerikus adat (Regressziós fa).

Ezt a modellt a következő szekcióban bemutatott Véletlen Erdő módszer miatt mutattam be, ugyanis az több döntési fát illeszt az adatokhoz és a végeredményt többségi szavazással dönti el.



2.3. ábra. A döntési fa működési elve.

A döntési fa előnyei:

- Nagyon könnyű megérteni és értelmezni. Az emberek túlnyomó többségének elég csak egyszer megmutatni ezt a módszert és szinte azonnal megértik. Ez azért van így, mert ez modellezi le legjobban az emberi döntés hozást.
- Sok adat esetén is kiválóan működik, a fa illesztése ugyan NP-nehéz probléma, de egy meglévő fát bejárni az attribútumhalmaz alapján elég gyors művelet. A hasznosságok, költségek, valószínűségek megismeréséhez sokszor van szükség külsős szakértőkre.
- Könnyen vehetünk fel új forgatókönyveket a modellünkhöz, hiszen bármikor tudjuk bővíteni azt. Ezután akár már több változóval rendelkező attribútumhalmazokat is megadhatunk bemenetnek.
- Segít megtalálni a legjobb, legrosszabb forgatókönyveket a hasznosságok és költségek segítségével, persze ehhez azokat ismerni kell.
- Ebben a szekcióban én fekete dobozként mutattam be, de ez alapvetően egy fehér doboz modell, hiszen pontosan tudjuk, hogy mit és hogyan csinál, és az egészet ábrázolni tudjuk egy fa gráfként.
- Könnyen kombinálható más döntéstámogató eszközökkel, hiszen akár a bemenetére, akár a kimenetére köthetünk egyet. Pl. az eredménye lehet egy neurális háló bemenete.

A döntési fa hátrányai:

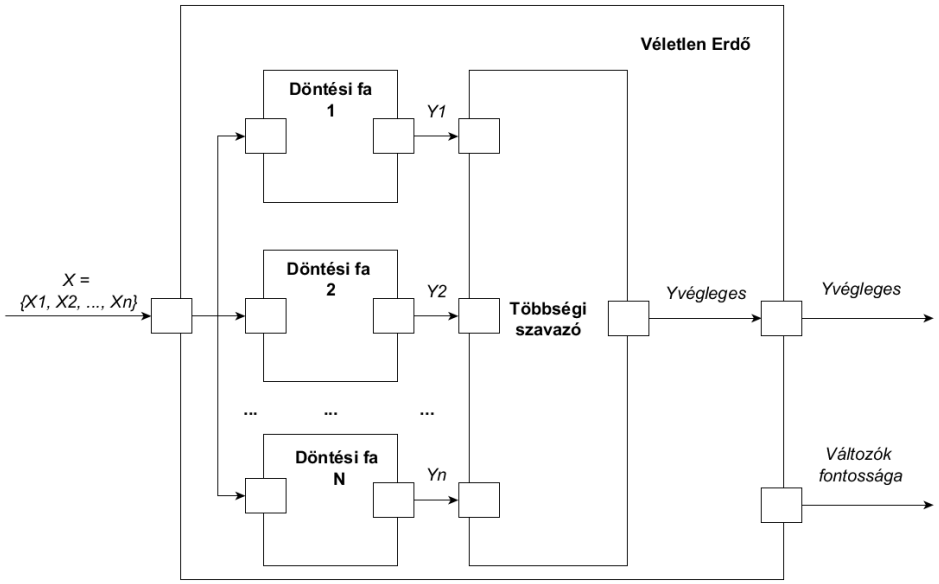
- Nem igazán robusztus modell, ami azt jelenti, hogy ha a bemenetének egyik változója akár egy picit is változik, akkor a kimenet drasztikusan más lehet.

- A túltanulás, mint a legtöbb tanítással előállított modellben itt is lehetséges. Az előbbi fogalom azt jelenti, hogy túl jól illeszkedik a tanuló példákra, ilyenkor nem tud általánosítani az illesztett modellünk, ezért a predikciós ereje nem feltétlenül helyes.
- A modell illesztése, egy NP-nehéz probléma, ezért a gyakorlatban különböző heurisztikus mohó algoritmusokat használnak.

További szakirodalmak [11], [13] a módszerről.

2.2.2. Véletlen Erdők

A *Véletlen Erdő* (**Random Forest** vagy **RF**) modell a döntési fák módszerét veszi alapul, de ez csökkenti a túltanulás kockázatát. Működése során a bemenete a benne található döntési fák bemenete lesz, azok külön-külön adnak egy eredményt, majd többségi szavazás segítségével megkapjuk a végső eredményt. A modell illesztése közben megkapjuk a változók fontossági értékeit is. Munkám során ezt a módszert használtam a rendszerből megkapott adatokon.

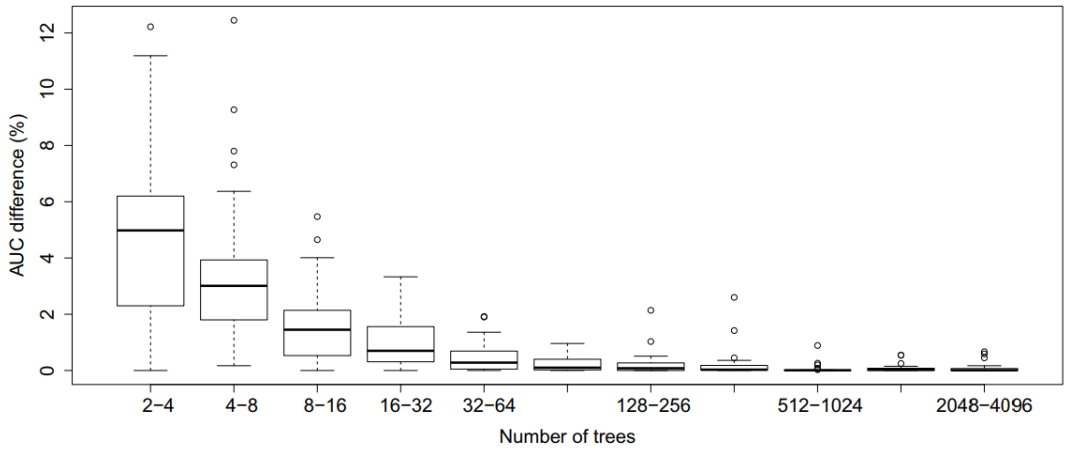


2.4. ábra. A Véletlen Erdő működési elve.

A Véletlen Erdő modell előnyei:

- Számos szakirodalom [13], [2], [14] az egyik legpontosabb gépi tanulási algoritmusnak tartja. Ezt több különböző, eltérő adathalmazon is bemutatták.
- A modell illesztése közben másodlagos eredményeként visszaadja a bemeneti változóink fontosságának sorrendjét, és klasszifikációs probléma esetén azt is, hogy az egyes osztályokba kerülést mely változók befolyásolták a legjobban.
- Az algoritmus a zsákolás (bagging) [19] módszerét veszi alapul, melynek lényege, hogy a modell létrehozásához az adatsoroknak csupán egy részhalmazát használja fel. A véletlen erdő modell illesztésekor az egyes szavazó fáknál véletlenszerűen kihagyott adatsorokat OOB soroknak nevezünk. Az egyes szavazási fák létrehozása után az OOB adatokkal leteszteli a fát és ennek a hiba arányoknak az összessége lesz az OOB hiba százalék.

- A Véletlen Erdő modellen nem szükséges kereszt validációt elvégezni, hiszen az OOB adatokkal történő tesztelés pontosan egy keresztvalidációnak felel meg.
- A nagy számok törvényével bizonyítható, hogy a fák számának növelése miatt nem tud túltanulni a modellünk. [2] Ennek ellenére egy szakirodalom [14] bemutatta több különböző adathalmazon is, hogy az erdőnek 64-128 fából érdemes állnia, mert úgy a legoptimálisabb az algoritmus sebessége és annak pontossága közötti arány.



2.5. ábra. A Véletlen Erdő fáinak száma és a pontosságok változása a tanulmány [14] szerint.

A Véletlen Erdő modell hátrányai:

- A legnagyobb hátránya a komplexitás és az algoritmus lassúsága, már egy döntési fa illesztése is lassú és ennél a modellben többet is létrehozunk belőle.
- Az előnyöknél említettem, hogy a fák számának növelése miatt nem fog túltanulni a modell, de ha a tanító adatkészletben túl sok kiugró érték van, a teszt adathalmazban pedig nem, akkor, mint a legtöbb másik gépi tanulási módszer, ez is túltanulhat.
- A döntési fa modell is lehet átláthatatlan, ez a probléma még inkább jellemző a Véletlen Erdőkre.

A Véletlen Erdő modellünk kétféle változó fontossági grafikonja:

- **Mean Decrease Accuracy (MDA) vagy %IncMSE**: Az előbbi nevet akkor használjuk, ha klasszifikációs problémához illesztettük a Véletlen Erdő modellünket, az utóbbit pedig regressziós esetben. Azt mutatja meg, hogy az egyes előrejelző változók értékeinek véletlenszerű permutálásával hogyan csökken a modell pontossága. Minél nagyobb egy változó MDA értéke, annál fontosabb.
- **Mean Decrease GINI (MDG) vagy %IncNodePurity**: Az előbbi nevet akkor használjuk, ha klasszifikációs problémához illesztettük a Véletlen Erdő modellünket, az utóbbit pedig regressziós esetben. Azt mutatja meg, hogy a döntési fák illesztésekor egy vágásnál mennyit csökkent a gini impurity értéke, regressziós fánál pedig mennyit nőtt az átlagos négyzetes hiba (MSE). Minél nagyobb egy változó MDA értéke, annál fontosabb.

Egy tanulmány [26] bebizonyította, hogy bizonyos előrejelző változók esetén mind a két metrika elfogult lehet, ezért külön-külön egyiket sem szabad használni, hanem a kettő segítségével együttesen kell kiszámolni azt, ez a szakirodalom [7] ad ey algoritmust arra, hogy ezt pontosan hogyan kell végrehajtani a gyakorlatban.

1. Illesszük az adatokhoz a Véletlen Erdő modellt és gyűjtsük össze minden változóra az MDA és MDG értékét.

2. Adjunk egy-egy új pontszámot a változóknak, az elérhető legnagyobb érték legyen a változók száma, a legkisebb pedig 1. Miután csökkenő sorrendbe rendeztük a változókat külön-külön az MDA és MDG értékek alapján már ki is oszthatjuk mindnek a két új pontszámát.

3. Adjuk össze minden változónak az előzőekben kapott két új pontszámát.

4. A változók csökkenő sorrendbe rendezése után megkapjuk a végleges változó fontossági sorrendet.

További szakirodalmak [1], [20], [8], [6] a módszerről.

2.2.3. Predikciós módszerek pontosságának mérőszámai

Ebben az alszekcióban azokat a metrikákat mutatom be, amelyek megadják, hogy egy Véletlen Erdő vagy Döntési Fa modell mennyire helyes. Minden adatelemezési módszernél az adatokat azért bontjuk külön tanító és tesztelő adathalmazra, hogy az előbbi segítségével illesszük a modellt, az utóbbival pedig megnézhessük annak helyességét, jelen dolgozatban a teljes adatkészlet egyik fele volt tanító, a másik pedig tesztelő adatkészlet, persze mindig véletlenszerűen válogattam ki a sorokat. A modellünk létrehozása után, bemenetként átadjuk neki a teszt adatkészletet, a modell minden sorra megmondja a tippjét, ezután az eredeti válasszal összehasonlítjuk és klasszifikációs probléma esetén megnézzük, hogy hány százalékban volt pontos, regressziós problémánál pedig általában a négyzetes hiba összeget (MSE).

A Véletlen Erdők módszeréhez az OOB hiba százalék egy fontos metrika, hiszen már illesztéskor kapunk egy becslést arra, hogy mennyire lesz pontos a modellünk. A konfúziós mátrix is egy fontos eszköz arra, hogy megbizonyosodhassunk a modellünk helyességéről, bár ezt csak klasszifikációs probléma esetén tudjuk használni. Ez megadja, hogy melyik osztálynál hányszor tippelt jóra a modellünk és azt is, hogy hányszor tippelt más osztályra és, hogy mi volt az. A konfúziós mátrixokhoz is elérhető számos metrika (F-measure, ROC, AUC), de ezeket csak két osztály esetén lehet használni, vagy a konfúziós mátrixok átalakításával, de jelen dolgozatban erre nem volt szükségem.

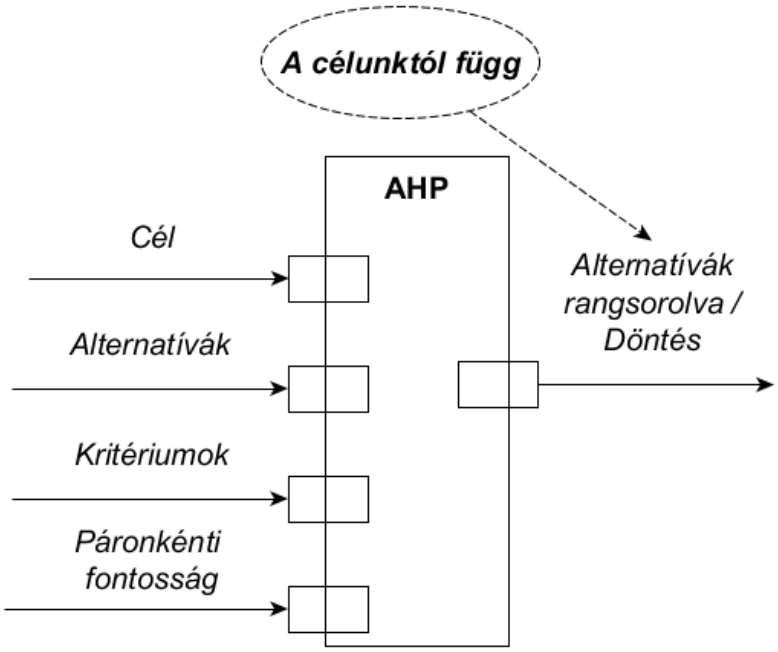
2.3. Döntéstámogató módszerek

Ebben a szakaszban bemutatom a manapság legtöbbször használt két döntéstámogatási módszert, melyek döntést hozni a használójának több alternatíva közül úgy, hogy a kritériumok száma igen nagy lehet és ezek akár ellent is mondhatnak egymásnak.

2.3.1. Analytic Hiearchy Process

Az *AHP* egy struktúrált döntéstámogatási módszer, amelyet Thomas L. Saaty fejlesztett ki az 1970-es években, azóta rengeteget finomítottak és fejlesztettek rajta. Az algoritmus nem a "korrekt" döntést adja meg, hanem azt, hogy a bemeneteink alapján melyik döntés

illik hozzánk a leginkább a céljainkat figyelembe véve, éppen ezért, ha változtatunk a bemenő paramétereken, akkor általában az eredményünk is változni fog.



2.6. ábra. Az AHP döntéstámogatási módszer működési elve.

A következők lennének a bemenetei:

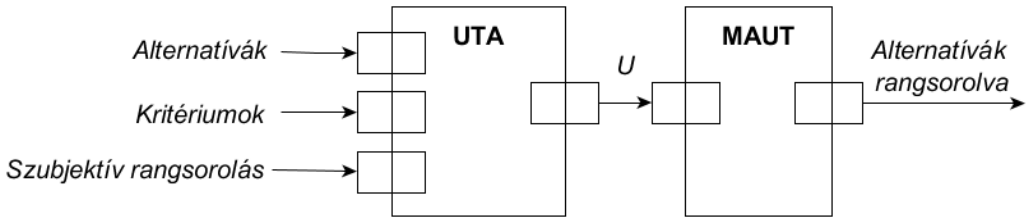
- Egy cél, amivel megmondjuk, hogy pontosan mit szeretnénk.
- A lehetséges alternatívák, amik között választani tudunk.
- A kritériumok, amelyek segítségével ki tudjuk értékelni az alternatíváinkat.
- Páronkénti prioritás a kritériumokra.

A módszer végeredménye, egy rangsorolás az alternatívákra, ez a bemenetek alapján jön létre az algoritmus lefutása után.

A témával kapcsolatos forrás [23], és két példa [28], [27], amelyek lépésről-lépésre bemutatják az algoritmus működését.

2.3.2. A Multi-attribute Utility Theory és az UTilitiés Additives módszerek

A többaspektusú döntéstámogatás az operációkutatás egyik altudománya. Lényege, hogy segít meghozni egy döntést, még akkor is, ha több különböző, egymásnak ellentmondó kritérium van. Én a Multi-attribute Utility Theory vagy más néven Multi-attribute Value Theory (**MAUT**) módszert használtam a munkám során. A módszer lényege, hogy az összes alternatíva minden kritériumához hozzárendel egy U hasznosság értéket, amely a fontosságot mutatja meg, az általam használt hasznosság hozzárendelő algoritmus az UTilitiés Additives (**UTA**) volt. A hasznosságok alapján megkapjuk a végleges eredményt, ami az alternatívák rangsorolása lesz.



2.7. ábra. A MAUT és az UTA módszerek működési elve.

További szakirodalmak [9], [5], [10], [25] a témával kapcsolatban.

2.4. Az esettanulmányról

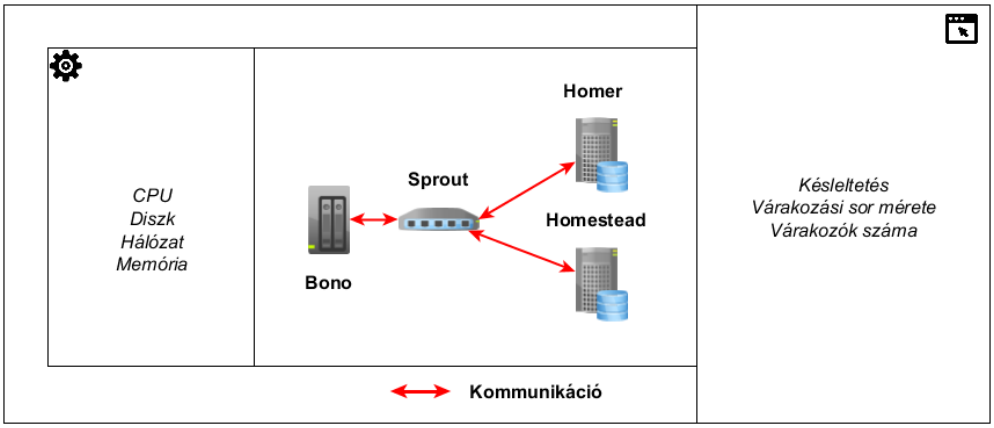
Először is arról az informatikai rendszerről írok, aminek a terhelésvizsgálata után a konzulensemék az általam felhasznált adatokat kapták. Utána röviden bemutatom az adatokat, amelyeken az elemzéseket végeztem, és amik segítettek a döntéstámogató program létrehozásában. Ezután a teljesítménymodellt, mint egy komponenst fogom bemutatni. Végezetül néhány mintakérdéseket fogok feltenni, melyet a rendszerünkhöz tehetünk fel.

2.4.1. Clearwater Projekt ismertetése

A Clearwater architektúra már az alapoktól úgy lett tervezve, hogy a virtualizált és a felhő rendszerek optimálisan fussanak rajta. Masszívan skálázható webes alkalmazások építésére és futtatására tervezték és arra, hogy illeszkedjen a SIP és IMS követelményeire, ezt a rengeteg tervezési minta segítségével tudja megtenni, amelyeket felhasznál. Az architektúra egy IP Multimedia Subsystem referencia implementáció, tehát megmondja, hogy hogyan/milyen elemekből kell felépíteni egy IMS alrendszert, illetve egyben akár egy ilyen rendszerként is használható. Az IP Multimedia Subsystem fogalmát Sharma a cikkjében így definiálja: [24] "egy szabvány, amely egy generikus architektúrát nyújt a Voice Over IP és egyéb multimédiás szolgáltatás számára".

Az OpenStack [17], [4], amely egy nyílt forráskódú felhő keretrendszer, segítségével privát vagy publikus felhőket tudunk létrehozni. Ez kiválóan alkalmas arra, hogy a Clearwater környezetet futtassa. A módszerem akár az OpenStack környezetbe is átvihető.
Az architektúra tulajdonságai:

- Minden komponens horizontálisan skálázható egy egyszerű, állapotmentes terhelésegyensúlyozó használatával.
- A hosszú életű állapotot egy erre használt csomópont a "Vellum" tárol, amit arra készítettek, hogy a felhő alapú tárolási technológiára (pl. Cassandra) optimalizáltak. Sehol máshol nem tároljuk el a hosszú életű állapotokat, így gyorsabban és könnyebben megy a dinamikus skálázódása a klasztereknek és ezzel minimalizáljuk a behatását annak, ha egy csomópont kiesik.
- A front-end SIP komponensei és a back-end szolgáltatásai közötti kezelőfelületek REST applikációs interfésszel rendelkező webes szolgáltatásokat használnak.
- A különböző komponensek közötti interfészek a kapcsolatok statisztikus újrahasznosítását és a connection pooling módszert használják arra, hogy biztosítsák a terhelés egyenlő elosztását még akkor is, ha új csomópontot adunk hozzá vagy veszünk el bármely rétegtől.

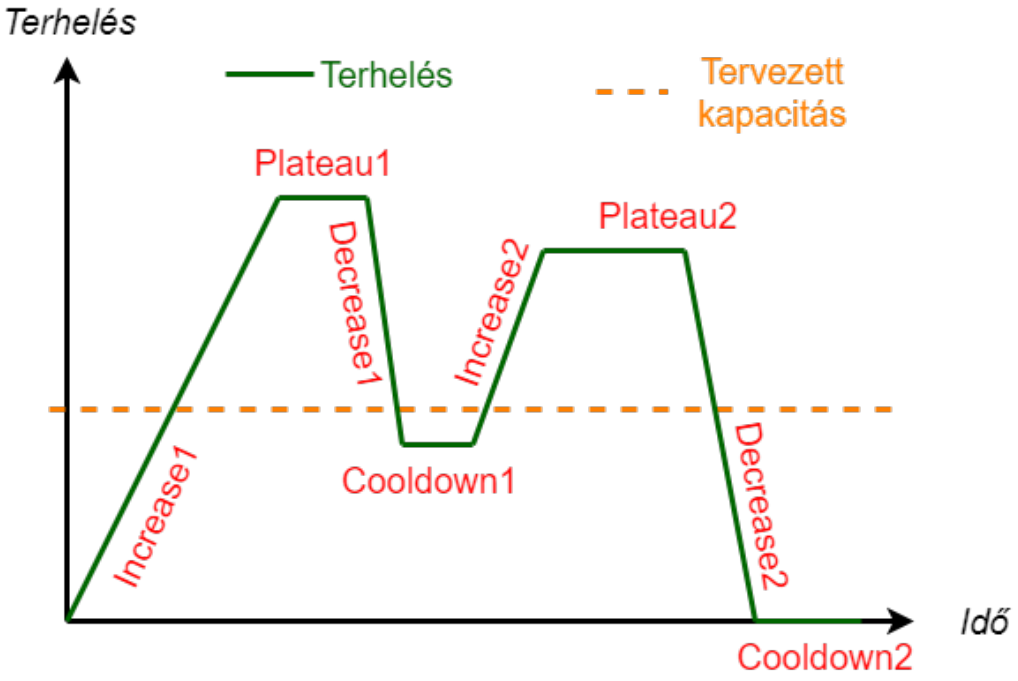


2.8. ábra. A Clearwater architektúra és annak komponensei [15] .

További források [15], [3] a projekttel [16] kapcsolatban.

2.4.2. Clearwater terheléstesztje

A MIT tanszék terheléstesztet hajtott végre egy olyan külsős cég rendszerén, amelynek felépítése azonos az előző szekcióban ismertetett architektúrával. A mérést kétféleképpen végezték el először úgy, hogy nem volt zavar a rendszerben *(Baseline mérés)*, majd azután úgy, hogy mesterséges interferenciát generáltak *(Interference mérés)*. A fent említett tesztet többször is végrehajtották.



2.9. ábra. Terhelés-idő grafikon.

A fenti ábra csupán tájékoztató jellegű, a tényleges Clearwater rendszerhez mérés után kapott terhelés-idő grafikon csupán kissé hasonlít erre. Két trapézjelet láthatunk, ezeknek külön-külön négy fázisa van:

- **Increase**: Ekkor a terhelés elkezd növekedni.

- **Plateau**: A növekedés után következő fázis, miután az elérte a maximum terhelést, akkor azt tartjuk egy ideig.
- **Decrease**: Ekkor újra elkezd csökkenni a rendszerben a terhelés.
- **Cooldown**: Hasonlít a plató fázisra, itt is egy konstans értéken tartjuk a terhelést.

2.4.3. Clearwater rendszerrel kapcsolatos korábbi kutatások

Számos nagyszerű tanulmányt [21], [22] találhatunk, amely a Clearwater teljesítménymodellével és terhelésével foglalkozik, ezek mind helyesek, de látható, hogy kevés túl dolgot mérnek, statisztikai elemzéseket használnak. Jelen dolgozatban egy pontosabb módszert fogok bemutatni, amely több mérés esetén is jól használható, a statisztikai elemzésnél erősebb adatelemzési algoritmusokat használ.

A korábban ismertetett AHP módszert is használják arra, hogy felhő rendszerek döntéstámogatási kérdéseire választ kapjanak. [12] Jelen tanulmányban én azért nem használom a döntéstámogatási módszereket, mert azt inkább olyan esetben kell, ha nem tudunk dönteni és külső segítségre van szükségünk, ráadásul számunkra adottak bizonyos siker metrikák és a cél is. Munkám során én a kategorizáló és előrejelző gépi tanulási algoritmusokat használtam és nem pedig alternatíva kiértékelő módszerekkel. A jövőben viszont, ha különböző skálázási alternatívák megjelennek, akkor már használhatóak lesznek az előzőekben ismertetett döntéstámogatási módszerek.

2.4.4. Az adatok bemutatása

Az adatok, amelyeket alapján a teljesítménymodellemet létrehoztam az előzőekben ismertetett mérés eredményei voltak. Összesen 14 darab .csv kiterjesztésű fájlt kaptam a munkámhoz (13498 sor), amelyek közül az utolsó 7 darabban volt interferencia. A sorok egymás után tartalmazzák a tesztek eredményét. Az *Interference* és a *Baseline* adatok is ugyanazokat az oszlopokat tartalmazzák, a különbség csupán annyi, hogy az előbbi még a zavarást szimuláló számítógépek adatait is tartalmazza.
Az adatok változóit öt csoportba lehet sorolni:

- **Platform.** Különböző adatok, amelyek a rendszer 4 komponenséből (Bono, Sprout, Homer, Homestead) származnak. Numerikus és kvantált formában is jelen vannak az adathalmazba, A teljesítménymodellemben a bemenetek közül a kapacitásnak, az erőforrásoknak és azok topológiájának feleltethetőek meg. A konkrét adathalmazban ide tartozik minden olyan oszlop, amely a "VM" kifejezéssel kezdődik, ezek a CPU, memória, diszk metrikák, folyamatok számának leírása.

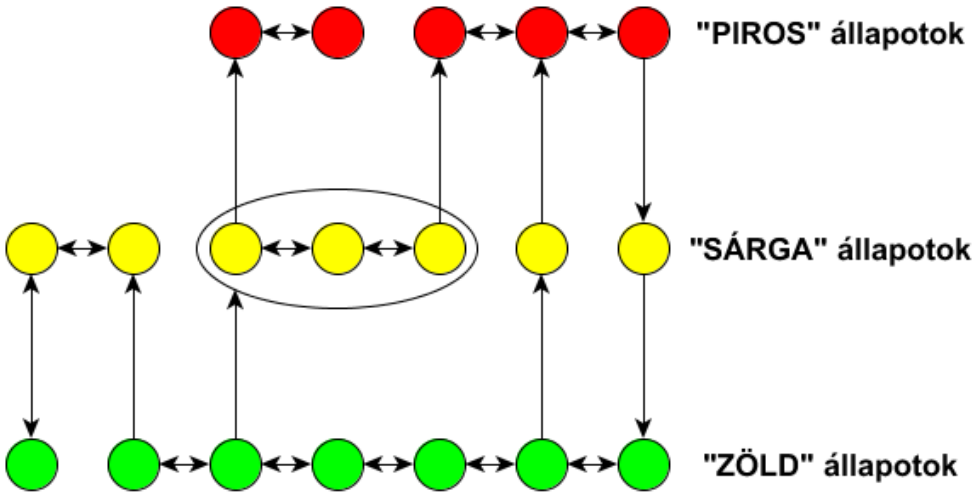
2.10. ábra. A Platform adatok jelölése a dolgozatomban.

- **SNMP**. A várakozási sor mérete, a rendszereben található kérések száma. A teljesítménymodellemben ezek az adatok felelnek meg a számítógép belső állapotainak (várakozási sor mérete, kérések átlagos száma). Az "APP" kifejezéssel kezdődő oszlopnevek tartoznak ehhez a kategóriához. Ezek az applikáció késleltetését, visszautasításait, kéréseik számát, várakozási sorának méret írják le.

2.11. ábra. Az SNMP adatok jelölése a dolgozatomban.

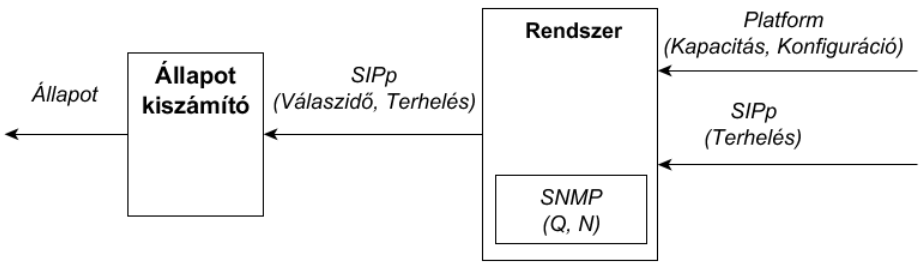
- **SIPp**. Regiszterek értékei, egy-egy folyamat felépítésének, lebontásának ideje, sikerek- és kudarcok száma, válaszidők, figyelmeztetések, hiba ráták, terhelések. A teljesítménymodellemben a terhelési adatok a bemenetek közül a terhelésnek felelnek meg, a többi pedig a rendszerünk kimenetei lesznek, ezen utóbbiak alapján lett generálva a rendszer állapot.

- **Állapotváltozók** A rendszer aktuális és következő állapotát írják le. A ZÖLD állapot azt jelenti, hogy a követelményeknek megfelelően működik a rendszer, a SÁRGA azt, hogy veszélyes állapotban van, a PIROS pedig, hogy megsértettük a követelményeket. A teljesítménymodellünkben ezek az értékek nem jelennek meg ezeket külső definíciók alapján generáljuk a rendszerhez idézett követelményekből (SIPp kimeneti adatok közül a válaszidő, a hibák és hívások rátája).



2.12. ábra. Példa a rendszerünk állapotterére.

- **Egyéb**. Innen egy fontosabb változót emelnék ki azt, amelyik megmutatja, hogy az adott sor mely terhelési fázisban volt. Ennek csak azért van jelentősége, mert az adatokat többféle szűrés mentén elemeztem, melyek közül az egyik éppen a fázisonkénti felbontás volt. Az időbélyeg változót is ebbe a csoportba sorolnám.



2.13. ábra. A teljesítménymodell a bemutatott adatokkal.

2.4.5. Mintakérdések és válaszaik

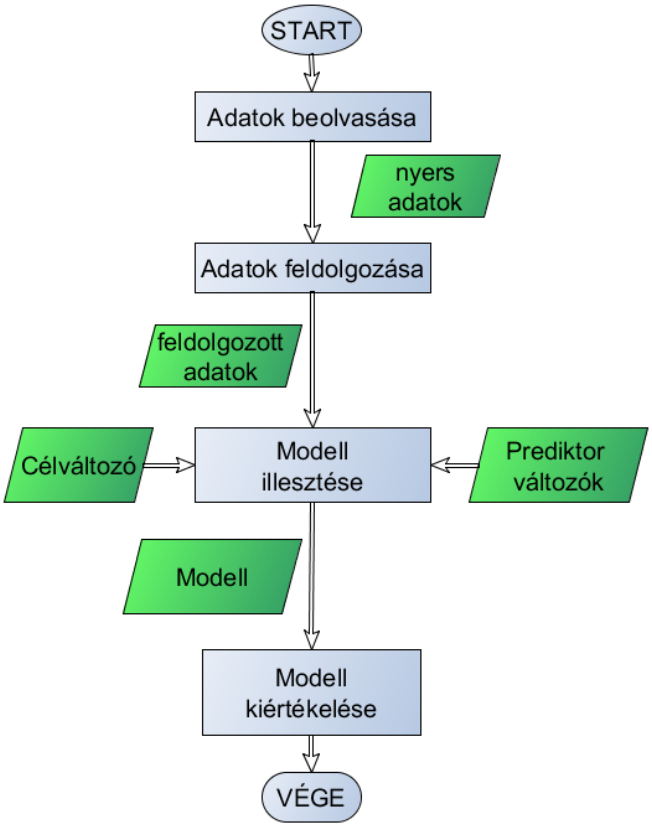
Néhány példa a rendszerhez feltehető mintakérdésekre, ezekre a válaszokat az olvasó majd a későbbi fejezetekben találhatja meg.

- **Mely Platform, befolyásolták a legjobban azt, hogy a rendszernek mi lesz az aktuális állapota?**
- **Mely SNMP, befolyásolták a legjobban azt, hogy a rendszernek mi lesz az aktuális állapota?**
- **Az interferencia mennyiben befolyásolja a végső állapotot?**
- **Mely metrikákat érdemes mérnem?**

3. fejezet

Felhő alapú infrastuktúra elemzése

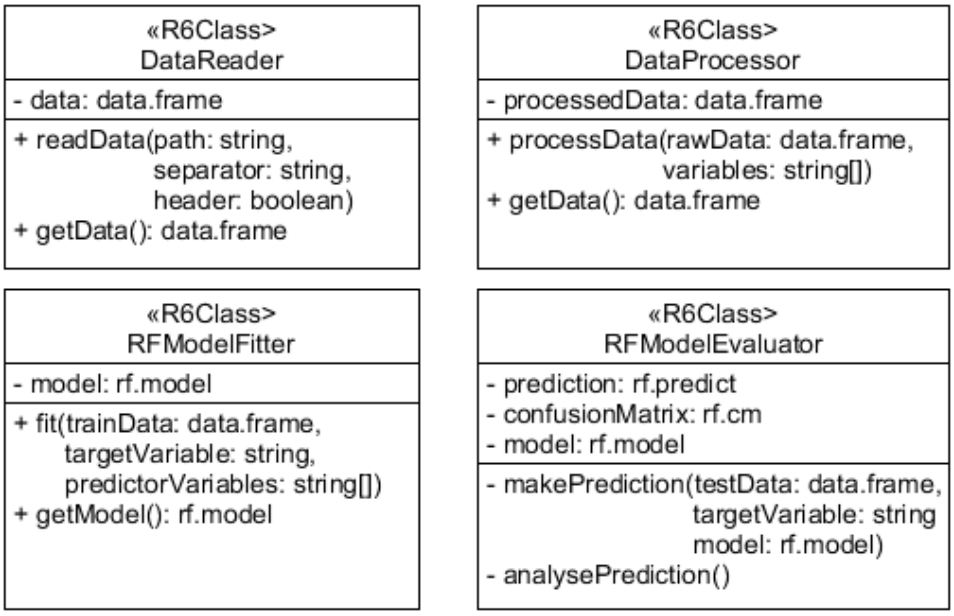
3.1. A módszerem ismertetése



3.1. ábra. A módszer folyamatmodellje.

Először is el kell döntenünk, hogy melyik adatkészletet szeretnénk használni (az interferenciával rendelkezőt vagy sem). Ezután a programom beolvassa a megfelelő adatkészletet, majd fel is dolgozza azokat, ez utóbbi abból áll, hogy kitörli azokat a sorokat, amelyek üres értéket tartalmaznak, valamint eldobja azokat a változókat, amelyeket nem szeretnénk felhasználni az elemzésünkhöz. A következő lépésben kiválasztjuk az előrejelző- és célváltozókat, majd felbontjuk az adathalmazt tanító és tesztelő adatkészletre. Ezután illesztjük a modellünket a tanító adathalmazhoz, így áll elő a Véletlen Erdő modellünk. Az illesztés után kiértékeljük a modellt a teszt adatok segítségével, hogy a modellünk mennyire helyes. Megkapjuk az OOB hiba százalékot, a konfúziós mátrixokat és a modellünk tesztelésének

eredménye után kiszámolt pontossági százalékot. Végezetül a módszerem kiszámolja azt, hogy mely változók voltak a legfontosabbak.



3.2. ábra. A programom osztálydiagramja.

A programomban 4 darab R6 osztályt készítettem el, ezek segítettek abban, hogy a fenti folyamatmodellt végrehajthassam.

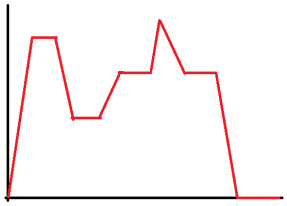
- A **DataReader** osztály azért felelős, hogy beolvasson és elmentsen egy megadott elérési úton levő fájlból az adathalmazunkat, amelyet a későbbiekben el is kérhetünk tőle további használatra.
- A **DataProcessor** osztály dolga, hogy ha kap egy adatot, akkor azt feldolgozza, ez pontosan azt jelenti, hogy kitörölje azokat a sorokat, amelyekben üres mező van és kiszedje a nem kívánatos oszlopokat.
- A **RFModelFitter** illeszt egy Véletlen Erdő modellt egy adathalmazhoz, megadhatjuk neki a cél- és előrejelző változók nevét. A létrehozott Véletlen Erdőt eltároljuk egy változóban, azért hogy a későbbiekben bármikor lekérhessük azt az osztálytól.
- A **RFModelEvaluator** felelőssége, hogy kiértékeljen egy Véletlen Erdő modellt bármely azzal kompatibilis teszthalmazon, ez tárolja el a már korábban említett konfúziós mátrixot és a tesztünk eredményét is.

3.2. A módszerem végrehajtása után kapott eredmények

Ebben a szekcióban megmutatom, hogy milyen eredmények jöttek ki, miután az előző fejezetben ismertetett adatokon végrehajtottam a módszeremet. Mint már korábban is említettem összesen 14 darab mérési eredmény volt, amiből csak az utolsó hétben volt jelen az interferencia. Három különböző mérést hajtottam végre, először is csak az adatok első felét használtam az adatoknak (Normál mérés), utána pedig az összeset (Normál és interferenciás mérés), végezetül pedig csak az utolsó hét darabot (Interferenciás mérés).

Ezt azért így csináltam, mert megakartam nézni, hogy a módszerem hogyan boldogul az interferencia jelenlétével.

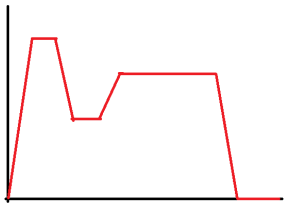
Fontos megjegyeznem, hogy elő lehet állítani olyan interferenciát, amelyet ez a rendszer nem tudna kezelni, de mivel ez egy az iparban használt rendszerek közül, ezért itt minél valósághűbb interferencia jelenik meg az adatokban, amelyek befolyásolják ugyan annak eredményét, de például nem rontják el teljesen azt.



3.3. ábra. Az Interferencia jelenlétének jelzése a dolgozatomban.

A célváltozó minden esetben a rendszer jelenlegi állapota volt, a prediktorok pedig először metrikusak voltak, utána pedig kvantáltak.
Az alábbi szűrési metrikákat használtam a méréseim során:

- **Minden adat(All Data):**
  Itt nem hajtottam végre az adatokon semmilyen szűrést.

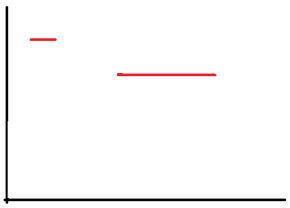


3.4. ábra. Az AllData mérés jelölése a dolgozatomban.

- **Emelkedő (Increase) fázis adatai:**
  Ebben az esetben csak azokat az adatokat használtam fel, amelyek az Increase fázishoz tartoznak.

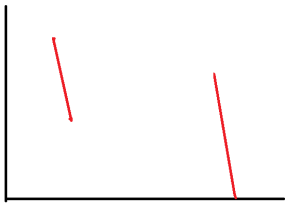
3.5. ábra. Az Increase mérés jelölése a dolgozatomban.

- **Plató (Plateau) fázis adatai:**
  Ebben az esetben csak azokat az adatokat használtam fel, amelyek az Plateau fázishoz tartoznak.



3.6. ábra. Az Plateau mérés jelölése a dolgozatomban.

- **Csökkenő (Decrease) fázis adatai:**
  Ebben az esetben csak azokat az adatokat használtam fel, amelyek az Decrease fázishoz tartoznak.



3.7. ábra. Az Decrease mérés jelölése a dolgozatomban.

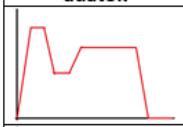
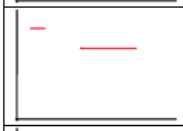
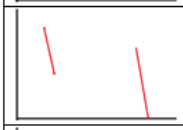
- **Lecsillapodó (Cooldown) fázis adatai:**
  Ebben az esetben csak azokat az adatokat használtam fel, amelyek az Cooldown fázishoz tartoznak.

3.8. ábra. Az Cooldown mérés jelölése a dolgozatomban.

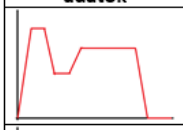
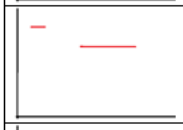
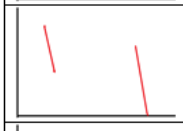
A fent említett szűréseket kétszer hajtottam végre az első esetben a prediktorok numerikus értékek voltak, a másodikban pedig kvantáltak.

3.2.1. Normál mérési eredmények

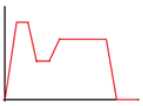
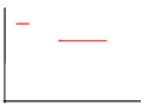
Ebben a részben az első hét adathalmazhoz illesztett Véletlen Erdők eredményeit mutatom be, először úgy, hogy a prediktor változók metrikusak voltak, majd utána pedig kvantáltak.

Mérés / Felhasznált adatok			
	OOB hiba: 23.41% Tesztelés pontossága: 77.37%	OOB hiba: 20.94% Tesztelés pontossága: 78.51%	OOB hiba: 27.34% Tesztelés pontossága: 73.41%
	OOB hiba: 11.35% Tesztelés pontossága: 86.02%	OOB hiba: 8.7% Tesztelés pontossága: 88.24%	OOB hiba: 8.11% Tesztelés pontossága: 84.95%
	OOB hiba: 30.91% Tesztelés pontossága: 71.1%	OOB hiba: 26.23% Tesztelés pontossága: 72.32%	OOB hiba: 35.93% Tesztelés pontossága: 65.25%
	OOB hiba: 28.77% Tesztelés pontossága: 75.12%	OOB hiba: 31.13% Tesztelés pontossága: 77%	OOB hiba: 35.38% Tesztelés pontossága: 69.95%
	OOB hiba: 7.62% Tesztelés pontossága: 92.62%	OOB hiba: 8.33% Tesztelés pontossága: 92.74%	OOB hiba: 8.45% Tesztelés pontossága: 91.43%

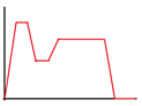
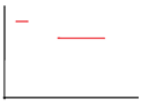
3.9. ábra. Normál mérés esetén az OOB hiba rátái és a tesztelés utáni pontosságai, metrikus prediktorok esetén.

Mérés / Felhasznált adatok			
	OOB hiba: 26.79% Tesztelés pontossága: 73.3%	OOB hiba: 41.43% Tesztelés pontossága: 60.47%	OOB hiba: 33.17% Tesztelés pontossága: 67.35%
	OOB hiba: 10.16 Tesztelés pontossága: 87.17	OOB hiba: 47.59% Tesztelés pontossága: 55.61%	OOB hiba: 9.63 Tesztelés pontossága: 86.1%
	OOB hiba: 33.63% Tesztelés pontossága: 65.48%	OOB hiba: 36.77% Tesztelés pontossága: 62.25%	OOB hiba: 42.2% Tesztelés pontossága: 58.31%
	OOB hiba: 37.74% Tesztelés pontossága: 67.14%	OOB hiba: 31.13% Tesztelés pontossága: 63.85%	OOB hiba: 35.85% Tesztelés pontossága: 60.56%
	OOB hiba: 8.57% Tesztelés pontossága: 92.26%	OOB hiba: 31.55% Tesztelés pontossága: 66.79%	OOB hiba: 7.98% Tesztelés pontossága: 91.9%

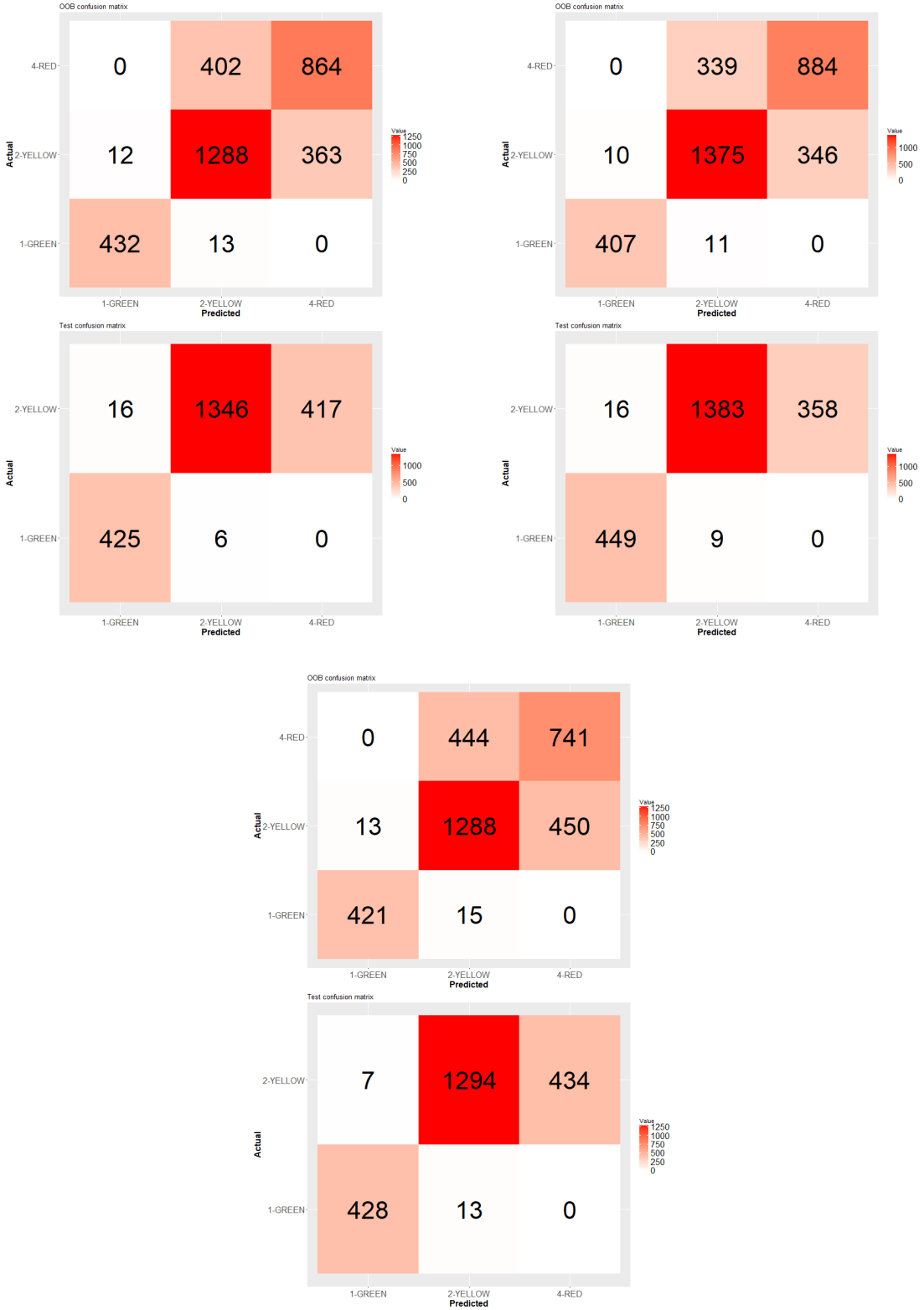
3.10. ábra. Normál mérés esetén az OOB hiba rátái és a tesztelés utáni pontosságai, kvantált prediktorok esetén.

Mérés / Felhasznált adatok			
	APP_LATHomercnt_Sprout_0 APP_LATlircnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 VM_CPUUtil_idle_Homestead_0 APP_LATlirvar_Sprout_0	VM_CPUUtil_user_Homestead_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_system_Homestead_0 VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_idle_Homestead_0	APP_Clients_Bono_0 APP_LATlircnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATlirvar_Sprout_0
	VM_CPUUtil_system_Homer_0 APP_LATavg_Bono_0 VM_CPUUtil_system_Bono_0 VM_CPUUtil_user_Homer_0 VM_CPUUtil_idle_Bono_0	VM_CPUUtil_usage_Bono_0 VM_CPUIntr_Bono_0 VM_CPUUtil_softirq_Homestead_0 VM_CPUUtil_user_Bono_0 VM_CPUUtil_idle_Homer_0	APP_LATmax_Sprout_0 APP_Clients_Bono_0 APP_LATvar_Bono_0 APP_INTRsuccessINVITE_Sprout_0 APP_LATliravg_Sprout_0
	VM_MEMavailable_Sprout_0 APP_Clients_Bono_0 VM_MEMavailable_Bono_0 VM_MEMfree_Bono_0 VM_CPUUtil_idle_Homestead_0	VM_MEMavailable_Sprout_0 VM_CPUUtil_user_Homestead_0 VM_CPUUtil_idle_Sprout_0 VM_CPUUtil_system_Homestead_0 VM_CPUUtil_idle_Bono_0	APP_Clients_Bono_0 APP_LATsarcnt_Sprout_0 APP_CACHELATcount_Homestead_0 APP_QUEcnt_Bono_0 APP_INTRattemptINVITE_Sprout_0
	APP_LATsarmin_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATlircnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATlirvar_Sprout_0	VM_CPUUtil_user_Homer_0 VM_CPUUtil_usage_Bono_0 VM_CPUUtil_user_Sprout_0 VM_CPUUtil_idle_Homer_0 VM_CPUUtil_softirq_Bono_0	APP_QUEvar_Bono_0 APP_REQin_Sprout_0 APP_LATHomesteadcnt_Sprout_0 APP_LATsarmin_Sprout_0 APP_QUEmax_Bono_0
	APP_LATlircnt_Sprout_0 APP_LATlirvar_Sprout_0 APP_LATHomercnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATmax_Homestead_0	VM_CPUUtil_user_Homestead_0 VM_CPUUtil_system_Homestead_0 VM_CPUUtil_system_Sprout_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_idle_Homestead_0	APP_LATlircnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATHomeravg_Sprout_0 APP_LATliravg_Sprout_0 APP_LATlirvar_Sprout_0

3.11. ábra. Normál mérés esetében az 5 legfontosabb változó, metrikus prediktorok esetén.

Mérés / Felhasznált adatok			
	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0	QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Sprout_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATmax_Sprout_0 QM_APP_LATsarvar_Sprout_0
	QM_APP_LATvar_Bono_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATmarvar_Sprout_0 QM_APP_LATsarvar_Sprout_0	QM_VM_CPUload5_Homestead_0 QM_VM_CPUload5_Sprout_0 QM_VM_CPUUtil_idle_Homestead_0 QM_VM_CPUUtil_idle_Homer_0 QM_VM_CPUUtil_iowait_Sprout_0	QM_APP_LATvar_Bono_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATmarvar_Sprout_0 QM_APP_LATuarvar_Sprout_0
	QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUUtil_idle_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_VM_CPUload5_Homer_0	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Homestead_0	QM_APP_LATlirvar_Sprout_0 QM_APP_LATmax_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATHomesteadmax_Sprout_0 QM_APP_LATHomervar_Sprout_0
	QM_APP_LATlirvar_Sprout_0 QM_VM_CPUload5_Bono_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATvar_Bono_0 QM_VM_CPUload5_Homer_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Sprout_0	QM_APP_LATlirvar_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATmax_Sprout_0 QM_APP_LATHomermax_Sprout_0
	QM_APP_LATsarvar_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATHomesteadvar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Sprout_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATHomesteadvar_Sprout_0 QM_APP_LATvar_Sprout_0 QM_APP_LATlirvar_Sprout_0

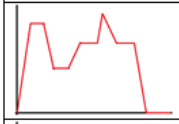
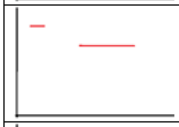
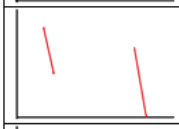
3.12. ábra. Normál mérés esetében az 5 legfontosabb változó, kvantált prediktorok esetén.



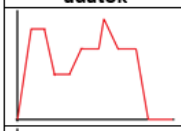
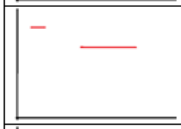
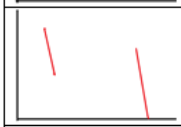
3.13. ábra. Normál mérésen belül az AllData-hoz tartozó konfúziós mátrixok, metrikus prediktorokkal. A sorrend megegyezik a fenti táblázattal (Platform és SNMP, Platform, SNMP).

3.2.2. Normál és interferencia mérési eredmények

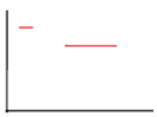
Ebben a részben mind a tizennégy adathalmazhoz illesztett Véletlen Erdők eredményeit mutatom be, először úgy, hogy a prediktor változók metrikusak voltak, majd utána pedig kvantáltak.

Mérés / Felhasznált adatok			
	OOB hiba: 22.22% Tesztelés pontossága: 77.52%	OOB hiba: 21.73% Tesztelés pontossága: 79.31%	OOB hiba: 27.83% Tesztelés pontossága: 73.11%
	OOB hiba: 9.16% Tesztelés pontossága: 93.03%	OOB hiba: 8.65% Tesztelés pontossága: 93.32%	OOB hiba: 9.65% Tesztelés pontossága: 90.03%
	OOB hiba: 27.68% Tesztelés pontossága: 70.91%	OOB hiba: 27.99% Tesztelés pontossága: 74.03%	OOB hiba: 35.88% Tesztelés pontossága: 65.34%
	OOB hiba: 26.82% Tesztelés pontossága: 69.88%	OOB hiba: 23.53% Tesztelés pontossága: 73.41%	OOB hiba: 33.88% Tesztelés pontossága: 66.59%
	OOB hiba: 6.61% Tesztelés pontossága: 92.32%	OOB hiba: 6.96% Tesztelés pontossága: 91.43%	OOB hiba: 8.39% Tesztelés pontossága: 91.79%

3.14. ábra. Normál és interferenciás mérés esetén az OOB hiba rátái és a tesztelés utáni pontosságai, metrikus prediktorok esetén.

Mérés / Felhasznált adatok			
	OOB hiba: 26.43% Tesztelés pontossága: 73.6%	OOB hiba: 38.48% Tesztelés pontossága: 60.72%	OOB hiba: 31.99% Tesztelés pontossága: 67.22%
	OOB hiba: 6.68% Tesztelés pontossága: 91.18%	OOB hiba: 38.77% Tesztelés pontossága: 60.96%	OOB hiba: 6.95% Tesztelés pontossága: 91.18%
	OOB hiba: 34.07% Tesztelés pontossága: 66.39%	OOB hiba: 37.03% Tesztelés pontossága: 64.03%	OOB hiba: 40.54% Tesztelés pontossága: 60%
	OOB hiba: 31.53% Tesztelés pontossága: 66.35%	OOB hiba: 31.53% Tesztelés pontossága: 64.94%	OOB hiba: 38.35% Tesztelés pontossága: 62.35%
	OOB hiba: 6.55% Tesztelés pontossága: 91.73%	OOB hiba: 38.39% Tesztelés pontossága: 62.56%	OOB hiba: 7.38% Tesztelés pontossága: 92.92%

3.15. ábra. Normál és interferenciás mérés esetén az OOB hiba rátái és a tesztelés utáni pontosságai, kvantált prediktorok esetén.

Mérés / Felhasznált adatok			
	APP_Clients_Bono_0 APP_LATlircnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATcount_Bono_0	VM_CPUUtil_user_Sprout_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_user_Homestead_0 VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_system_Sprout_0	APP_Clients_Bono_0 APP_INTRattemptINVITE_Sprout_0 APP_LATlircnt_Sprout_0 APP_LATcount_Bono_0 APP_LATlirmax_Sprout_0
	VM_CPUUtil_system_Homestead_0 APP_LATHomercnt_Sprout_0 APP_LATHomermin_Sprout_0 APP_LATHomervar_Sprout_0 APP_LATHomeravg_Sprout_0	VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_idle_Homer_0 VM_CPUUtil_softirq_Bono_0 VM_CPUUtil_usage_Sprout_0	APP_REQin_Homestead_0 APP_LATlircnt_Sprout_0 APP_LATmax_Bono_0 APP_Clients_Bono_0 APP_LATlirmax_Sprout_0
	APP_Clients_Bono_0 VM_MEMavailable_Sprout_0 APP_QUEcnt_Bono_0 VM_CPUIntr_Bono_0 APP_REQin_Bono_0	VM_CPUUtil_user_Homestead_0 VM_CPUUtil_usage_Sprout_0 VM_MEMavailable_Sprout_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_system_Homestead_0	APP_Clients_Bono_0 APP_CACHELATcount_Homestead_0 APP_LATcount_Bono_0 APP_QUEcnt_Bono_0 APP_LATsarcnt_Sprout_0
	APP_Clients_Bono_0 APP_INTRattemptINVITE_Sprout_0 VM_CPUUtil_user_Bono_0 APP_LATHomercnt_Sprout_0 APP_QUEmax_Bono_0	VM_CPUUtil_user_Bono_0 VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_system_Homestead_0 VM_CPUUtil_idle_Sprout_0 VM_CPUUtil_user_Sprout_0	APP_INTRattemptINVITE_Sprout_0 APP_LATHomermax_Sprout_0 APP_Clients_Bono_0 APP_LATHomercnt_Sprout_0 APP_CACHELATcount_Homestead_0
	APP_LATHomercnt_Sprout_0 APP_LATlircnt_Sprout_0 APP_LATlirmax_Sprout_0 APP_LATliravg_Sprout_0 APP_LATHomermax_Sprout_0	VM_CPUUtil_system_Homestead_0 VM_CPUUtil_user_Homestead_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_softirq_Sprout_0	APP_LATHomermin_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATlircnt_Sprout_0 APP_LATHomeravg_Sprout_0

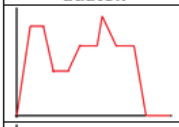
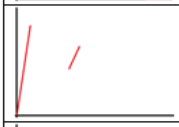
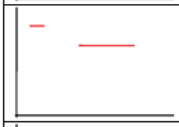
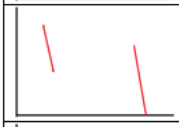
3.16. ábra. Normál és interferenciás mérés esetében az 5 legfontosabb változó, metrikus prediktorok esetén.

Mérés / Felhasznált adatok			
	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0 QM_APP_LATvar_Bono_0 QM_VM_CPUUtil_iowait_Sprout_0	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Homestead_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATmax_Sprout_0 QM_APP_LATsarvar_Sprout_0
	QM_APP_LATHomervar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATmarvar_Sprout_0 QM_APP_LATsarvar_Sprout_0	QM_VM_CPUload5_Homestead_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Sprout_0 QM_VM_CPUUtil_idle_Bono_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATmarvar_Sprout_0 QM_APP_LATuarvar_Sprout_0
	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_APP_LATvar_Bono_0 QM_APP_LATmax_Sprout_0	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Homestead_0	QM_APP_LATvar_Bono_0 QM_APP_LATmax_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATsarmax_Sprout_0 QM_APP_LATHomervar_Sprout_0
	QM_VM_CPUload5_Bono_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_APP_LATmax_Sprout_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUload5_Sprout_0	QM_APP_LATlirvar_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATmax_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATHomesteadmax_Sprout_0
	QM_APP_LATHomervar_Sprout_0 QM_APP_LATHomesteadvar_Sprout_ 0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATvar_Sprout_0 QM_APP_LATlirvar_Sprout_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUload5_Homestead_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATvar_Sprout_0 QM_APP_LATHomesteadvar_Sprout_0 QM_APP_LATlirvar_Sprout_0

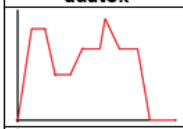
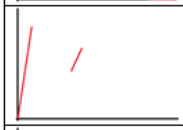
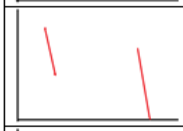
3.17. ábra. Normál és interferenciás mérés esetében az 5 legfontosabb változó, kvantált prediktorok esetén.

3.2.3. Interferencia mérési eredmények

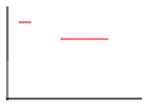
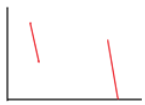
Ebben a részben az utolsó hét adathalmazhoz illesztett Véletlen Erdők eredményeit mutatom be, először úgy, hogy a prediktor változók metrikusak voltak, majd utána pedig kvantáltak.

Mérés / Felhasznált adatok			
	OOB hiba: 21.43% Tesztelés pontossága: 79.49%	OOB hiba: 20.52% Tesztelés pontossága: 79.94%	OOB hiba: 26.59% Tesztelés pontossága: 74.25%
	OOB hiba: 4.81% Tesztelés pontossága: 97.85%	OOB hiba: 2.14% Tesztelés pontossága: 95.7%	OOB hiba: 4.28% Tesztelés pontossága: 96.24%
	OOB hiba: 27.17% Tesztelés pontossága: 72.79%	OOB hiba: 26.37% Tesztelés pontossága: 73.49%	OOB hiba: 34.29% Tesztelés pontossága: 65.67%
	OOB hiba: 28.3% Tesztelés pontossága: 74.65%	OOB hiba: 31.13% Tesztelés pontossága: 78.4%	OOB hiba: 31.6% Tesztelés pontossága: 68.08%
	OOB hiba: 8.45% Tesztelés pontossága: 93.33%	OOB hiba: 7.98% Tesztelés pontossága: 92.98%	OOB hiba: 7.98% Tesztelés pontossága: 92.38%

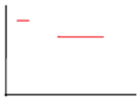
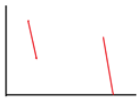
3.18. ábra. Interferenciás mérésnél az OOB hiba rátái és a tesztelés utáni pontosságai, metrikus prediktorok esetén.

Mérés / Felhasznált adatok			
	OOB hiba: 25.55% Tesztelés pontossága: 74.7%	OOB hiba: 37.02% Tesztelés pontossága: 61.99%	OOB hiba: 31.06% Tesztelés pontossága: 69.16%
	OOB hiba: 1.6% Tesztelés pontossága: 94.65%	OOB hiba: 36.9% Tesztelés pontossága: 67.91%	OOB hiba: 3.21% Tesztelés pontossága: 96.26%
	OOB hiba: 33.63% Tesztelés pontossága: 66.89%	OOB hiba: 35.18% Tesztelés pontossága: 64.96%	OOB hiba: 39.11% Tesztelés pontossága: 59.86%
	OOB hiba: 31.13% Tesztelés pontossága: 67.14%	OOB hiba: 33.96% Tesztelés pontossága: 63.85%	OOB hiba: 30.19% Tesztelés pontossága: 67.61%
	OOB hiba: 7.14% Tesztelés pontossága: 92.5%	OOB hiba: 37.62% Tesztelés pontossága: 62.74%	OOB hiba: 6.19% Tesztelés pontossága: 91.55%

3.19. ábra. Interferenciás mérésnél az OOB hiba rátái és a tesztelés utáni pontosságai, kvantált prediktorok esetén.

Mérés / Felhasznált adatok			
	APP_Clients_Bono_0 APP_LATlircnt_Sprout_0 VM_CPUUtil_user_Homestead_0 APP_LATcount_Bono_0 VM_CPUUtil_usage_Sprout_0	VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_user_Sprout_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_idle_Sprout_0	APP_Clients_Bono_0 APP_INTRattemptINVITE_Sprout_0 APP_LATcount_Bono_0 APP_LATlircnt_Sprout_0 APP_LATHomercnt_Sprout_0
	VM_CPUUtil_idle_Homer_0 VM_CPUUtil_iowait_Sprout_0 VM_CPUUtil_system_Sprout_0 APP_LATcount_Bono_0 APP_REQin_Bono_0	VM_CPUUtil_idle_Bono_0 VM_CPUUtil_idle_Homer_0 VM_CPUUtil_idle_Sprout_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_user_Sprout_0	APP_QUEavg_Bono_0 APP_QUEvar_Bono_0 APP_LATliravg_Sprout_0 APP_LATmax_Bono_0 APP_LATvar_Bono_0
	APP_REQin_Bono_0 VM_MEMavailable_Sprout_0 APP_Clients_Bono_0 APP_LATsarcnt_Sprout_0 VM_CPUUtil_system_Homer_0	VM_CPUUtil_system_Homer_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_idle_Sprout_0 VM_CPUUtil_user_Homestead_0 VM_MEMavailable_Sprout_0	APP_Clients_Bono_0 APP_CACHELATcount_Homestead_0 APP_LATsarcnt_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATHomesteadcnt_Sprout_0
	APP_LATHomercnt_Sprout_0 VM_CPUUtil_user_Bono_0 APP_Clients_Bono_0 APP_LATmin_Sprout_0 APP_CACHELATcount_Homestead_0	VM_CPUUtil_user_Bono_0 VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_system_Homestead_0 VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_system_Sprout_0	APP_Clients_Bono_0 APP_REQin_Sprout_0 APP_LATlircnt_Sprout_0 APP_LATHomervar_Sprout_0 APP_LATavg_Bono_0
	VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_idle_Homestead_0 APP_LATHomeravg_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATlirmin_Sprout_0	VM_CPUUtil_idle_Homestead_0 VM_CPUUtil_user_Homestead_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_system_Sprout_0	APP_Clients_Bono_0 APP_LATHomervar_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATlircnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0

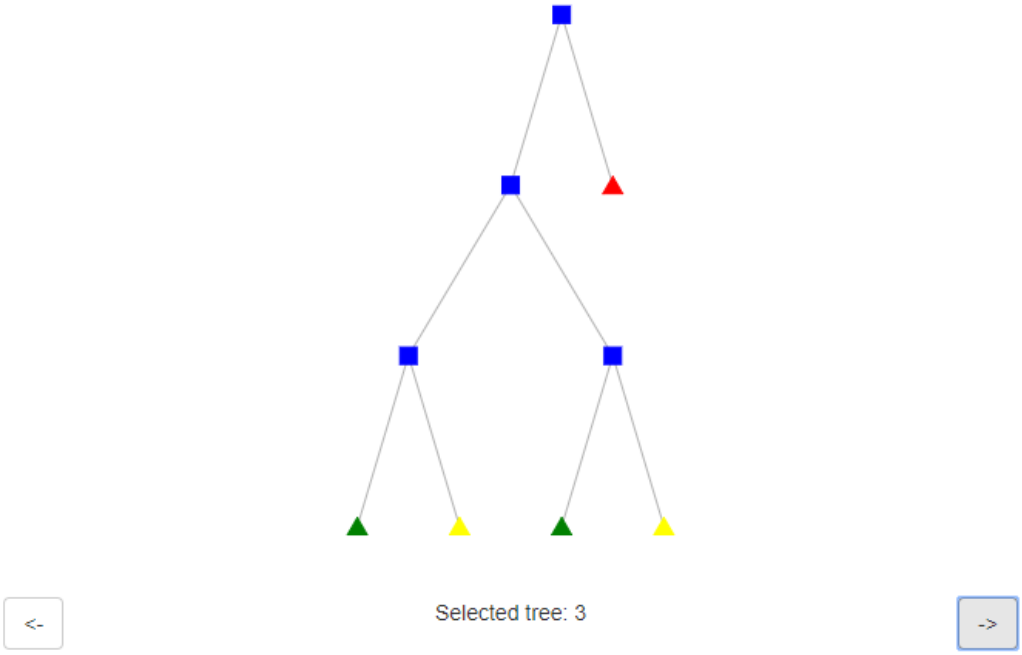
3.20. ábra. Interferenciás mérés esetében az 5 legfontosabb változó, metrikus prediktorok esetén.

Mérés / Felhasznált adatok			
	QM_APP_LATHomervar_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_APP_LATvar_Bono_0	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUUtil_idle_Interference_0 QM_VM_CPUload5_Homer_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATmax_Sprout_0 QM_APP_LATvar_Sprout_0
	QM_APP_LATvar_Bono_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATuarvar_Sprout_0 QM_APP_LATsarvar_Sprout_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUload5_Sprout_0 QM_VM_CPUUtil_idle_Bono_0 QM_VM_CPUUtil_idle_Sprout_0	QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATuarvar_Sprout_0 QM_APP_LATmarvar_Sprout_0
	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUUtil_idle_Interference_0	QM_APP_LATmax_Sprout_0 QM_APP_LATHomermax_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATsarmax_Sprout_0 QM_APP_LATHomesteadmax_Sprout_0
	QM_APP_LATlirvar_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_APP_LATHomervar_Sprout_0 QM_APP_LATvar_Bono_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUUtil_idle_Bono_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATmax_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATsarmax_Sprout_0
	QM_APP_LATHomervar_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATHomesteadvar_Sprout_0 QM_APP_LATvar_Bono_0 QM_APP_LATvar_Sprout_0	QM_VM_CPUload5_Bono_0 QM_VM_CPUload5_Homer_0 QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUUtil_idle_Bono_0	QM_APP_LATHomervar_Sprout_0 QM_APP_LATHomesteadvar_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATlirvar_Sprout_0 QM_APP_LATvar_Sprout_0

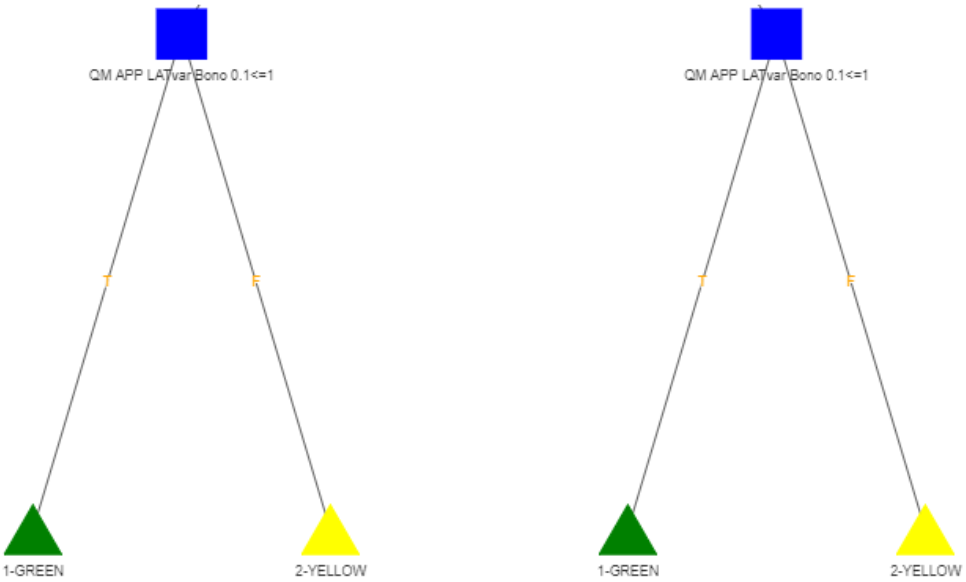
3.21. ábra. Interferenciás mérés esetében az 5 legfontosabb változó, kvantált prediktorok esetén.

3.3. Véletlen Erdő modell megjelenítő

Az R programozási nyelvhez elég sok döntési fa megjelenítő csomagot találhatunk. A véletlen erdőkhöz is van, de sajnos ezek eléggé átláthatatlan formában ábrázolják azt. Ezért munkám során írtam egy saját, interaktív Véletlen Erdő megjelenítőt, amely képes egy fájlba elmentett modellt beolvasni és átlátható gráfos formában megjeleníteni azt. Az interaktivitás abban valósul meg, hogy minden csomópontot tudunk horizontálisan mozgatni, vagy a felhasználó akár teljesen bele is közelíthet a gráfba. Minden illesztés után kimentettem az eredményt egy **.rf.RDS** kiterjesztésű állományba. Ez lehetővé teszi, hogy egy-egy fa belső szerkezetét megvizsgáljuk.



3.22. ábra. A Véletlen Erdő modell megjelenítő.



3.23. ábra. A Véletlen Erdő modell megjelenítő képes beleközelíteni a megjelenített fába.

A fenti ábrán egy részletet vágtam ki az egyik Véletlen Erdő modelljének fájából. A kék négyzetek a vágási csomópontoknak felelnek meg, a háromszögek pedig a levelek / lehetséges kimenetei a fának. A csomópontok színei és formái testre szabhatóak.

4. fejezet

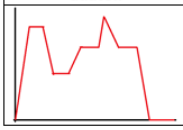
A módszer kiértékelése

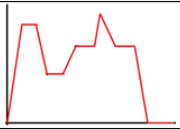
4.1. Érdemes-e külön véletlen erdő modellt építeni minden terhelési fázisra?

Az előző fejezetben található OOB hiba rátákon és pontosságokon jól látszik, hogy nem érdemes minden terhelési szakaszhoz külön-külön Véletlen Erdő modellt építeni, hanem elég csak egyet, és azt felhasználni az előrejelzések során, ezek azt is megmutatják, hogy a modellem jó maradhat, ha az adatok karakterisztika változatlan marad, de a terhelés változik.

4.2. A módszerem kiértékelése váratlan interferencia esetén

A teljesítménymodellem akkor is elég jól teljesít, ha olyan adatokhoz illesztek Véletlen Erdő modellt, amelyekben nem volt jelen az interferencia, de esetleg a modell előrejelző képességének használata közben mégis megjelent volna ez a rendszerben. Ezt szimulációval egy újabb mérés segítségével próbáltam ki, melyben a Baseline adatokhoz illesztett Véletlen Erdőt egy olyan tesztelő adathalmazon próbáltam ki, melyben jelen volt az interferencia. A lenti ábrán jól látható, hogy a modellem ekkor is helyes maradt.

Mérés / Felhasznált adatok			
	OOB hiba: 21.36% Tesztelés pontossága: 77.45%	OOB hiba: 19.7% Tesztelés pontossága: 72.05%	OOB hiba: 27.44% Tesztelés pontossága: 77.1%

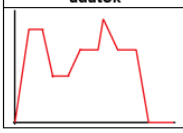
Mérés / Felhasznált adatok			
	APP_Clients_Bono_0 VM_CPUUtil_user_Homestead_0 APP_INTRattemptINVITE_Sprout_0 APP_LATHomercnt_Sprout_0 APP_LATlircnt_Sprout_0	VM_CPUUtil_system_Homestead_0 VM_CPUUtil_softirq_Sprout_0 VM_CPUUtil_user_Homestead_0 VM_CPUUtil_usage_Homestead_0 VM_CPUUtil_system_Sprout_0	APP_Clients_Bono_0 APP_LATHomercnt_Sprout_0 APP_INTRattemptINVITE_Sprout_0 APP_LATlircnt_Sprout_0 APP_LATsarcnt_Sprout_0

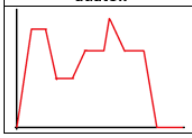
4.1. ábra. A modell kipróbálása nem várt interferencia esetén.

Ennek a mérésnek a segítségével két fontos tanulságot sikerült megtalálnom, az első, hogy további mérések szükségesek a rendszer szélsőértékeinek megtaláláshoz. A második pedig, hogy a rendszer mostani mérései nem használják ki elég jól a rendszert, ezt onnan lehet látni, hogy az interferencia szinte semmilyen befolyásolással sincs a modellem pontosságára.

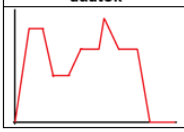
4.3. A módszer használata a rendszerben található komponensekre

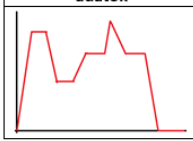
Azt is megszeretném mutatni, hogy a módszerem a rendszert felépítő komponenseken is használható, igaz ebben az esetben új Véletlen Erdő kell, de ha valamilyen új komponens kerül a rendszerbe, akkor a módszerem segítségével ahhoz is könnyen tudok új Véletlen Erdő modellt illeszteni. Erre is szeretnék mutatni egy példát, legyen most a célváltozónk a Bono Platformon található késleltetésnek az eloszlása **(QM__APP__LATvar__Bono__0)**, ez a változó is a ZÖLD, SÁRGA, PIROS értékek egyikét veheti fel. Mivel a Bono csak a Sprout útválasztón keresztül tud kommunikálni, ezért nézzük meg, hogy mi történik, ha az SNMP, illetve a Bono és a Sprout komponensekhez tartozó Platform metrikák segítségével szeretnénk előrejelezni a késleltetés eloszlását. Ehhez illesztettem egy új Véletlen Erdő modellt az adatokhoz.

Mérés / Felhasznált adatok			
	OOB hiba: 0.33% Tesztelés pontossága: 99.59%	OOB hiba: 1.48% Tesztelés pontossága: 98.49%	OOB hiba: 0.19% Tesztelés pontossága: 99.61%

Mérés / Felhasznált adatok			
	APP_QUEvar_Bono_0 APP_QUEcnt_Bono_0 APP_LATcount_Bono_0 APP_LATcount_Sprout_0 APP_QUEmax_Bono_0	VM_CPUUtil_softirq_Sprout_0 VM_CPUIntr_Bono_0 VM_CPUUtil_idle_Sprout_0 VM_CPUUtil_system_Sprout_0 VM_CPUUtil_user_Sprout_0	APP_REQin_Bono_0 APP_LATcount_Sprout_0 APP_QUEvar_Bono_0 APP_QUEcnt_Bono_0 APP_LATcount_Bono_0

4.2. ábra. A modell kipróbálása a Bono komponensre, metrikus prediktorokkal.

Mérés / Felhasznált adatok			
	OOB hiba: 3.76% Tesztelés pontossága: 95.89%	OOB hiba: 15.1% Tesztelés pontossága: 85.27%	OOB hiba: 4.02% Tesztelés pontossága: 96.21%

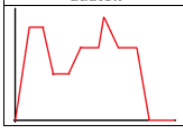
Mérés / Felhasznált adatok			
	QM_APP_LATlirvar_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATvar_Sprout_0 QM_APP_LATmarvar_Sprout_0 QM_APP_LATuarvar_Sprout_0	QM_VM_CPUUtil_idle_Sprout_0 QM_VM_CPUload5_Bono_0 QM_VM_CPUUtil_idle_Bono_0 QM_VM_CPUUtil_iowait_Sprout_0 QM_VM_CPUload5_Sprout_0	QM_APP_LATlirvar_Sprout_0 QM_APP_LATsarvar_Sprout_0 QM_APP_LATmarvar_Sprout_0 QM_APP_LATvar_Sprout_0 QM_APP_LATuarvar_Sprout_0

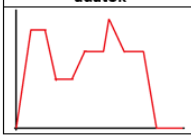
4.3. ábra. A modell kipróbálása a Bono komponensre, kvalitatív prediktorokkal.

Mint látható a komponensekre is helyes a modellem, sikerült megtalálni azt, hogy szinte 100%-ban a Sprout komponens és az applikációs adatok felel azért, hogy a Bono késleltetésének eloszlása jelenleg milyen értéket vehet fel.

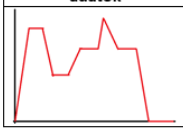
A Sprout rendszerre is megnéztem, hogy igaz-e, hogy a késleltetésének eloszlása **(QM__APP__LATvar__Sprout__0)** szoros kapcsolatban áll a vele összekötött Homer, illetve Homestead rendszerrel. Ehhez prediktor változóknak a Homer, Homestead, vala-

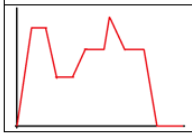
mint a Sprout Platform, illetve SNMP metrikáit vettem. Ebben az esetben is sikerült a módszeremnek megtalálnia a szoros kapcsolatot a komponensek között.

Mérés / Felhasznált adatok			
	OOB hiba: 0.04% Tesztelés pontossága: 99,88%	OOB hiba: 0,71% Tesztelés pontossága: 99, 44%	OOB hiba: 0.07% Tesztelés pontossága: 99.87%

Mérés / Felhasznált adatok			
	"APP_LATmax_Sprout_0" "APP_LATvar_Sprout_0" "APP_LATavg_Sprout_0" "APP_LATmax_Homestead_0" "APP_LATmin_Homestead_0"	"VM_CPUUtil_system_Homestead_0" "VM_CPUIntr_Homestead_0" "VM_CPUUtil_softirq_Sprout_0" "VM_CPUUtil_usage_Homestead_0" "VM_CPUUtil_user_Homestead_0"	"APP_LATvar_Sprout_0" "APP_LATmax_Sprout_0" "APP_LATavg_Sprout_0" "APP_CACHELATcount_Homestead_0" "APP_CACHELATavg_Homestead_0"

4.4. ábra. A modell kipróbálása a Sprout komponensre, metrikus prediktorokkal.

Mérés / Felhasznált adatok			
	OOB hiba: 0,52% Tesztelés pontossága: 99,66%	OOB hiba: 8.52% Tesztelés pontossága: 91,71%	OOB hiba: 0,39% Tesztelés pontossága: 99,56%

Mérés / Felhasznált adatok			
	"QM_APP_LATvar_Sprout_0.1" QM_APP_LATHomesteadvar_Sprout_0 "QM_APP_LATsarvar_Sprout_0" "QM_APP_LATlirvar_Sprout_0" "QM_APP_LATHomervar_Sprout_0"	"QM_VM_CPUUtil_idle_Sprout_0" "QM_VM_CPUload5_Homer_0" "QM_VM_CPUUtil_idle_Homer_0" QM_VM_CPUUtil_iowait_Sprout_0" "QM_VM_CPUload5_Sprout_0"	"QM_APP_LATvar_Sprout_0.1" "QM_APP_LATHomesteadvar_Sprout_0" "QM_APP_LATsarvar_Sprout_0" QM_APP_LATHomervar_Sprout_0" "QM_APP_LATlirvar_Sprout_0"

4.5. ábra. A modell kipróbálása a Sprout komponensre, kvalitatív prediktorokkal.

4.4. A módszer használata előrejelzésre

Eddig munkám során főként klasszifikációs problémákat oldottam meg, ugyanis a célváltozó az aktuális állapot volt, éppen ezért a továbbiakban predikciót fogok végrehajtani a modellem segítségével. Éppen ezért az adatokhoz hozzáadtam egy Y nevű változót, amely egy olyan struktúra, amely a jelenlegi és a következő állapotot jelöli (például [SÁRGA, PIROS]), azaz most azt néztem meg, hogy a modellem megtudja-e becsülni, hogy milyen állapotátmenet lesz a Platform és SNMP adatokat figyelve, a prediktorok a szokásosak voltak, ám ehhez még hozzávettem azt is, hogy mi a rendszerem aktuális állapota.

Ennél a mérésnél a precízió növelésének érdekében másképpen illesztettem a Véletlen Erdő modellt, korábban már bemutattam, hogy az MDA és az MDG kombinációjával hogyan lehet megkapni egy precízebb változó fontossági sorrendet, ezt az algoritmust itt használtam. Minden illesztés után meghagytam azokat a prediktor változókat, amely a rangsor felső 50%-ban voltak és ezt addig ismételtem, amíg a pontosság el nem kezdett csökkenni, ugyanis minden alkalommal, amikor kivettem néhány prediktort, amely sokszor a téves választ okozza a modellem pontossága megnőtt. A végére mindössze 27 maradt, amelyek különböző SNMP és Platform adatok, valamint az aktuális állapot.

OOB hiba ráta	Pontosság
23,98%	77,16%

4.1. táblázat. Állapotátmenetek előrejelzésének pontossága.

Legfontosabb változók
State
APP_LATlirvar_Sprout_0
SW_CurrentCall
APP_Clients_Bono_0
APP_LATmax_Sprout_0
VM_CPUIntr_Homer_0
VM_CPUIntr_Homestead_0
APP_LATHomercnt_Sprout_0
APP_LATlircnt_Sprout_0
APP_LATHomeravg_Sprout_0
APP_QUEmin_Bono_0
APP_PRRvar_Sprout_0
APP_QUEmin_Sprout_0
APP_REQin_Sprout_0
APP_QUEcnt_Bono_0
APP_REQin_Bono_0
APP_REQin_Homestead_0
VM_CPUCtxsw_Homestead_0
APP_LATHomermin_Sprout_0
APP_LATHomervar_Sprout_0
APP_LATlirmax_Sprout_0
APP_REJ_Bono_0
APP_LATcount_Homestead_0
VM_CPUUtil_system_Homestead_0
APP_REJ_Sprout_0
APP_REJ_Homestead_0
VM_CPUIntr_Bono_0

4.2. táblázat. Az állapotátmeneteket legjobban befolyásoló változók.

Mint látható a teszt adatok körülbelül 77% pontossággal megtudta mondani a modellem, hogy milyen állapotátmenet lesz. Ráadásul ezt úgy, hogy az interferencia is jelen volt a rendszeremben. Éppen ezért mint látható az általam elkészített teljesítménymodell előrejelzésre is kiválóan alkalmas. A második táblázat szépen összefoglalja, hogy mely változók befolyásolják az állapotátmeneteket. Fontos megjegyeznem, hogy egy típusú adott állapotátmenetre, pl. (ZÖLD, SÁRGA) nem tudjuk, hogy helyes-e a sorrend, ez minden lehetséges típusú átmenethez (lsd. 2.12 ábra) tartozik.

4.5. A SIPp terheléses adatok befolyásának vizsgálata

Korábban már említettem, hogy a SIPp terheléssel kapcsolatos adatait nehéz mérni és emiatt sokszor nem is áll rendelkezésünkre, éppen ezért végeztem egy olyan mérést is, ahol ezek az adatok is a prediktorok között voltak. Mint ahogyan a lenti táblázatban is látható nem sokkal lett pontosabb az eredmény, így ebből következtethetünk arra, hogy a SIPp adatokat nem érdemes mérnünk, mert sokkal költségesebb a mérése, mint az általa megnövelt pontossági százalék.

OOB hiba ráta	Pontosság
22,77%	78,41%

4.3. táblázat. A Véletlen Erdő OOB hiba rátája és pontossága abban az esetben, ha a SIPp terheléses adatok is a prediktorok között vannak.

Fontos megjegyeznem, ha irreálisan megterhelném a rendszert, akkor a modellem érvényét vesztené, de a gyakorlatban használt modern rendszerekben található terhelés elosztó és hozzáférés védelem biztosító nem engedné, hogy ez megtörténjen, így ettől az esettől nem kell tartanom. Ha a rendszerben valami változik, akkor új Véletlen Erdő modellt kell illeszteni az adatokhoz, melyhez szükség van a megváltozott adatokra is.

4.6. Válaszok a korábbi mintakérdésekre

Két fejezettel ezelőtt feltettem néhány mintakérdést a rendszerrel kapcsolatban, a következőkben ezekre szeretnék választ adni.

- **Mely Platform, befolyásolták a legjobban azt, hogy a rendszernek mi lesz az aktuális állapota?**
  A korábbi fejezetekben felsorolt legfontosabb változók között egyértelműen látszik, hogy a különböző rendszerek CPU kihasználtságával kapcsolatos változók dominálnak, így a Platform kategóriába eső változók közül ezek befolyásolják a legjobban a jelenlegi állapotot.

- **Mely SNMP, befolyásolták a legjobban azt, hogy a rendszernek mi lesz az aktuális állapota?**
  A korábbi fejezetekben felsorolt legfontosabb változók között egyértelműen látszik, hogy a különböző rendszerek késleltetéssel kapcsolatos változók dominálnak, így az SNMP kategóriába eső változók közül ezek befolyásolják a legjobban a jelenlegi állapotot.

- **Az interferencia mennyiben befolyásolja a végső állapotot?**
  Több olyan sort is találtam az adathalmazban, amely szerepel az interferencia nélküli és az azzal rendelkező adatok között, ezek közül több olyan eset is volt, amely a Baseline esetben SÁRGA állapotban volt, az Interference mérésnél mégis PIROS állapotba került, hála az interferenciának.

- **Mely metrikákat érdemes mérnem?**
  A változók fontosságát tekintve láthatjuk, hogy a SIPp terheléses adatok nem befolyásolják annyira a rendszerünk fontosság, és mivel azokat amúgy is nehéz mérni, ezért azt nem érdemes. A Platform és az SNMP adatok külön-külön és együtt is elég jól meghatározzák, hogy a rendszerünk mely állapotban van, így ezeket mindenképpen érdemes. A Platform osztályon belül az egyes CPU kihasználtsági metrikák a legbefolyásolóbbak, így azokat mindenképpen érdemes, az SNMP kategóriába eső változók közül a késleltetéses adtok a legbefolyásosabbak, így azokat is mindenképpen érdemes mérni.

A teljesítménymodellem végig követte azt, hogy nem volt állapotátmenet sem ZÖLD-ből PIROS-ba, sem pedig fordítva. Kiemelhető, hogy a jelenlegi kvantálást a felhasználók visszajelzései és a tapasztalatok alapján lettek elkészítve, emiatt lehetséges az, hogy a változók fontossága sokszor eltér numerikus és kvantált esetben.

5. fejezet

Összefoglalás

Munkám során bemutattam, hogy a mai modern adatelemzési és döntéstámogatási módszerek segítségével hogyan lehet magas szintű teljesítménymodelleket alkotni egy informatikai rendszerhez az abból kinyert mérési adatok segítségével. A módszerem azt is bemutatja, hogy a rendszerekben mely adatokat érdemes összegyűjtenünk, illetve mérnünk, a lényeges szűk keresztmetszetek is felderíthetjük a használatával, a segítségével komplex állapotokat tudunk becsülni és akár előrejelezni azt, hogy mi lesz a rendszer következő állapota.

A felhasználók bizonyos a rendszerrel kapcsolatos kérdéseire is választ tud adni. Például mely Platform metrikák befolyásolták a legjobban azt, hogy a rendszernek mi lesz az aktuális állapota.

Összefoglaltam a gyakorlatban legtöbbször használt predikciós, osztályozó és döntéstámogatási módszereket. Az elkészült teljesítménymodellem hatékonyságát egy valós esettanulmány segítségével mutattam be. A dolgozatom jelentős hangsúlyt fektet a rendszer kiértékelésére, melynek segítségével a későbbiekben akár vezérelni is tudjuk azt, illetve válaszokat nyújt a felhasználók bizonyos kérdéseire a rendszerrel kapcsolatban.

A későbbiekben egy olyan módszer után fogok kutatni, amely a Véletlen Erdő modellben található döntési fák csomópontjainál szereplő vágási feltételek alapján egy új, jobb kvantálást adhat az egyes változókra. A Véletlen Erdő modell megjelenítőmet is szeretném tovább fejleszteni, hogy segítségével bizonyos szempontok szerint lehessen elemezni a Véletlen Erdő modelljeinket. Szeretnék írni egy grafikus felhasználói felülettel rendelkező programot, amely automatizáltan elvégzi helyettem a megfelelő modell illesztéseket. Illetve újabb gépi tanulási algoritmusokat is kiszeretnék próbálni, a rendszeren, hátha akkor még egy ennél is pontosabb teljesítménymodellt tudok készíteni a rendszerhez.

Ábrák jegyzéke

Táblázatok jegyzéke

Irodalomjegyzék

[1] Leo Breiman: Random forests - classification description. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

[2] Leo Breiman: Random forests. *Mach. Learn.*, 45. évf. (2001. október) 1. sz., 5–32. p. ISSN 0885-6125. URL https://doi.org/10.1023/A:1010933404324. 28 p.

[3] Cseh Dávid: Kritikus felhőalkalmazások szolgáltatásbiztonságának kockázat vezérelt kiértékelése. Tdk-dolgozat (Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Kar). 2015.

[4] Zilahi Dávid: Privát felhők szolgáltatásbiztonságra optimalizálása és modell-vezérelt terítése. Tdk-dolgozat (Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Kar). 2015.

[5] James Dyer – José Figueira – Salvatore Greco – Matthias Ehrogott: Maut — multi-attribute utility theory. *International Series in Operations Research & Management Science Multiple Criteria Decision Analysis: State of the Art Surveys*, 2005. 01., 265–292. p.

[6] John Ehrlinger: ggrandomforests: random forests for regression. *arXiv preprint arXiv:1501.07196*, 2016.

[7] Hong Han – Xiaoling Guo – Hua Yu: Variable selection using mean decrease accuracy and mean decrease gini based on random forest, 2016. 08.

[8] Satoshi Hara – Kohei Hayashi: Making tree ensembles interpretable. *arXiv preprint arXiv:1606.05390*, 2016.

[9] Alessio Ishizaka – Philippe Nemery: *Multi-attribute utility theory.* 2013, Wiley, 81–113. p. ISBN 9781118644898.
URL http://dx.doi.org/10.1002/9781118644898.ch4.

[10] E. Jacquet-Lagreze – J. Siskos: Assessing a set of additive utility functions for multicriteria decision-making, the uta method. *European Journal of Operational Research*, 10. évf. (1982) 2. sz., 151–164. p. URL https://EconPapers.repec.org/RePEc:eee:ejores:v:10:y:1982:i:2:p:151-164.

[11] Craig W Kirkwood: Decision tree primer, 2002.

[12] Young-Chan Lee – Hanh Tang – Vijayan Sugumaran: A deployment model for cloud computing using the analytic hierarchy process and bcor analysis.

[13] Gilles Louppe: *Understanding Random Forests: From Theory to Practice.* PhD értekezés (University of Liege, Belgium). 2014. 10. arXiv:1407.7502.

[14] Thais Mayumi Oshiro – Pedro Santoro Perez – José Baranauskas: How many trees in a random forest? *Lecture notes in computer science*, 7376. évf. (2012. 07).

[15] Metaswitch Network: Clearwater architecture - project clearwater 1.0 documentation. http://clearwater.readthedocs.io/en/stable/Clearwater_Architecture.html.

[16] Metaswitch Network: Project clearwater. http://www.projectclearwater.org/.

[17] Openstack open source cloud computing software. https://www.openstack.org/.

[18] Gergely János Paljak – Imre Kocsis – Zoltán Égel – Dániel Tóth – András Pataricza: *Sensor Selection for IT Infrastructure Monitoring*. Berlin, Heidelberg, 2010, Springer Berlin Heidelberg, 130–143. p. ISBN 978-3-642-11482-3.
URL https://doi.org/10.1007/978-3-642-11482-3_9.

[19] J Ross Quinlan és mások: Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1* (konferenciaanyag). 1996, 725–730. p.

[20] Predrag Radenković: Random forest. home.etf.rs/~vm/os/dmsw/Random%20Forest.pptx/.

[21] Raphael Vicente Rosa – Claudio Bertoldo – Christian Esteve Rothenberg: Take your VNF to the gym: A testing framework for automated NFV performance benchmarking. *IEEE Communications Magazine*, 55. évf. (2017) 9. sz., 110–117. p.
URL https://doi.org/10.1109/MCOM.2017.1700127.

[22] Raphael Vicente Rosa – C Rothenberg: Taking open vswitch to the gym: An automated benchmarking approach.

[23] Thomas L. Saaty – Michael C. Fu: *Analytic Hierarchy Process*. Boston, MA, 2013, Springer US, 52–64. p. ISBN 978-1-4419-1153-7.
URL https://doi.org/10.1007/978-1-4419-1153-7_31.

[24] Madhu J. Sharma – Victor CM Leung: Ip multimedia subsystem authentication protocol in lte-heterogeneous networks. *Human-centric Computing and Information Sciences*, 2. évf. (2012. Oct) 1. sz., 16. p. ISSN 2192-1962.
URL https://doi.org/10.1186/2192-1962-2-16.

[25] Yannis Siskos – Evangelos Grigoroudis – Nikolaos F. Matsatsinis: Uta methods. *International Series in Operations Research & Management Science Multiple Criteria Decision Analysis: State of the Art Surveys*, 2005., 297–334. p.

[26] Carolin Strobl – Anne-Laure Boulesteix – Achim Zeileis – Torsten Hothorn: Bias in random forest variable importance measures: Illustrations, sources and a solution." bmc bioinformatics, 8(1), 25. *BMC bioinformatics*, 8. évf. (2007. 02), 25. p.

[27] Wikipedia: Analytic hiearchy processing – car example - wikipedia. https://en.wikipedia.org/wiki/Analytic_hierarchy_process_%E2%80%93_car_example/.

[28] Wikipedia: Analytic hiearchy processing – leader example - wikipedia. https://en.wikipedia.org/wiki/Analytic_hierarchy_process_%E2%80%93_leader_example/.