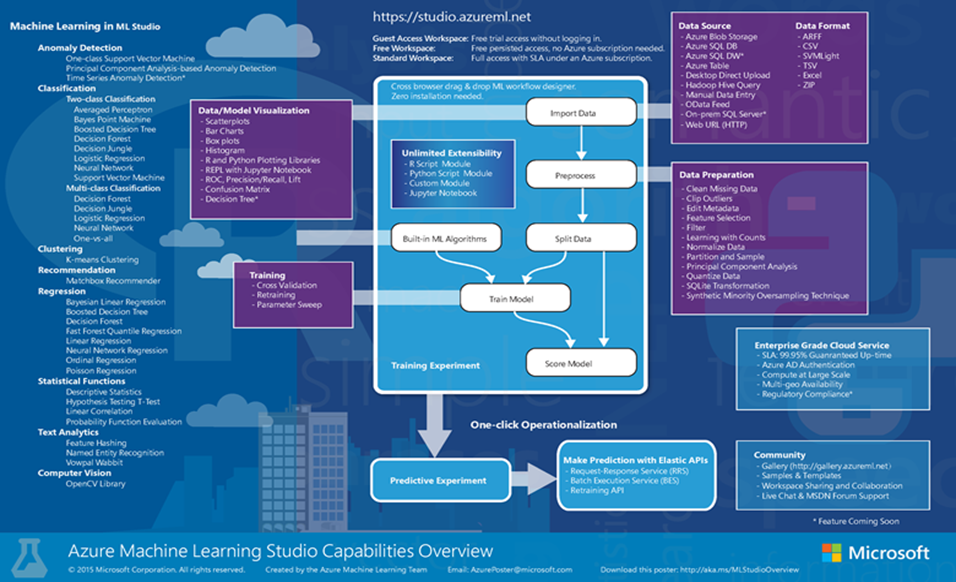
# Azure Machine Learning Studio

### Térkép



### Adatbázis feltöltési lehetőségek

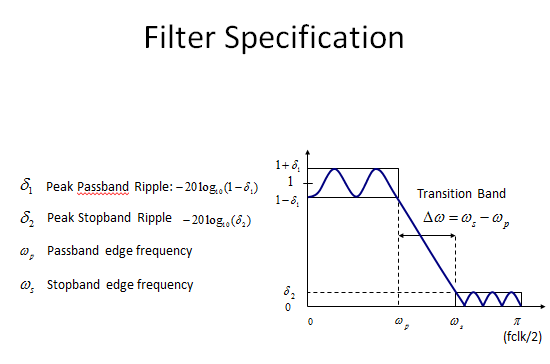
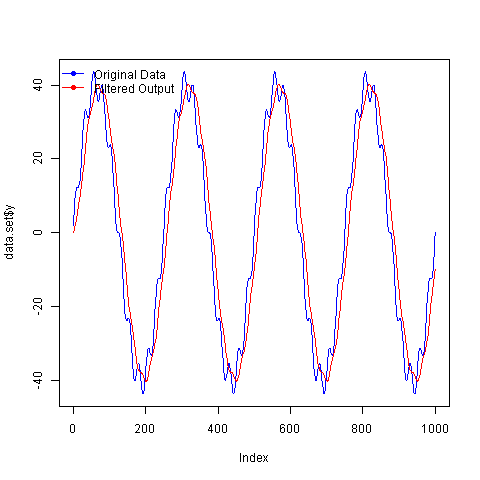
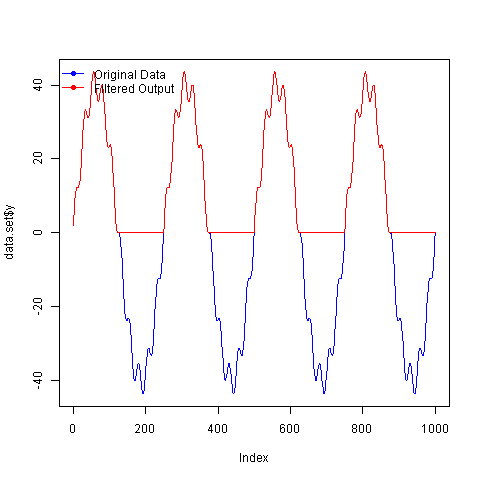
Az alábbi formátumban lehet adatszetteket feltölteni: .txt, .tsv, .csv, Excel fájl, Azure tábla, Sql adatbázis fájl, .svmlight, .arff, .Rdata. Továbbá, zip fájlt (.zip, .gz) is fel lehet tölteni.

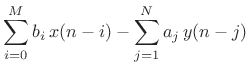
Hiányosságnak elkönyvelhető, hogy a csv fájlok esetében a pontosvesszőt elválasztásra használó fájlokat nem tudja kezelni a felület, a pontosvesszővel elválasztott oszlopokat egy oszlopnak tekinti.

## Adatok előkészítése

### Filterek / Szűrők

Elérhető szűrőtípusok és azok állítható paraméterei:

* 1. FIR filter
     1. Order: a szűrő foka, együtthatóinak száma (múltbeli bemeneti érték tényezők száma)
     2. Window: különböző ablakoló eljárások Hamming, Blackman, Triangular, Rectangular
     3. Filter type: low pass (aluláteresző), high pass (felüláteresztő), bandpass, bandstop
     4. Low cutoff
  2. IIR filter
     1. Order: a szűrő foka, együtthatóinak száma
     2. Filter kind: Butterworth, Chebysev
     3. Filter type
     4. Low cutoff
     5. Ripple
  3. Medián filter
  4. Moving average filter - Mozgóátlag szűrő
     1. 
  5. Threshold - Küszöbérték filter
     1. 
  6. Felhasználó által definiált filter
     1. Lehet FIR vagy IIR filter

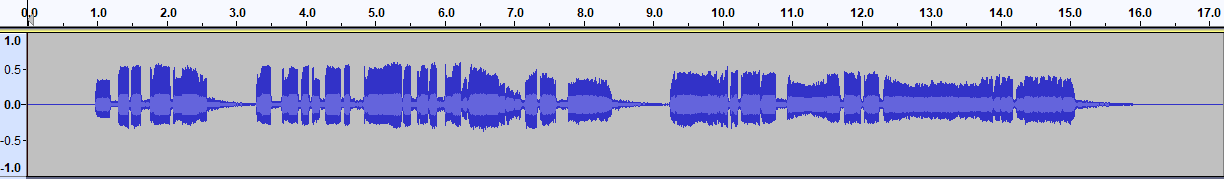


* + 1. FIR szűrő esetén meg kell adni a feed-forward (b) együtthatókat
    2. IIR szűrő esetén feed-forward (b) és feed-backward (a) együtthatók megadása szükséges ([IIR/FIR filters - Stanford](https://ccrma.stanford.edu/~jos/filters/Difference_Equation_I.html))
    3. Az együtthatók száma határozza meg a szűrő rendjét. Tetszőleges rendű szűrő adható meg.

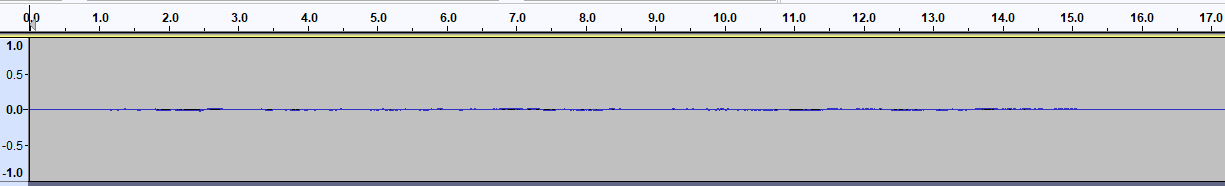
Példa projekt (Azure team): [link](https://gallery.azure.ai/Experiment/0f5e582eb9ee4d9d9f34126f593e840f).

### Audió fájlok filterezése

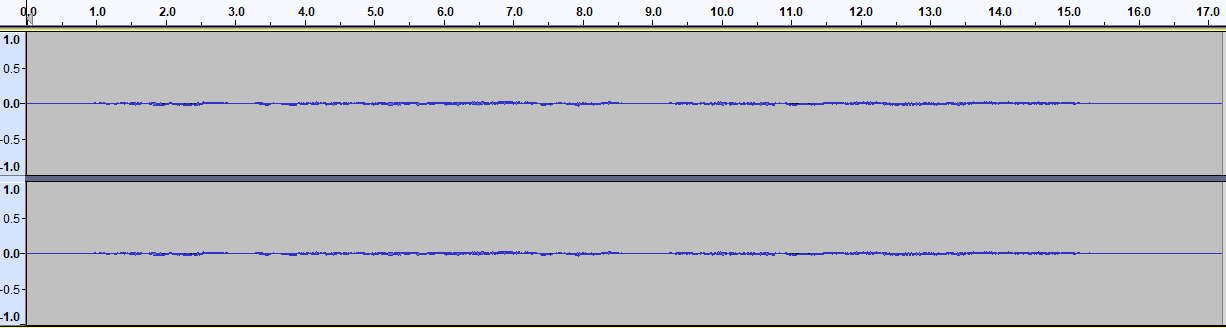
A szűrő modulok kipróbálása érdekében létrehoztam egy saját kísérletet, melyben egy rövid (17s) hangfájlon próbáltam ki a medián és egyéb filtereket. Az egyszerűség kedvéért a wav fájlt egy külső Python szkripttel csv formátumba alakítottam a *scipy.io.wavfile* csomag függvénye segítségével. A fájl 17.2 másodperc hosszú, mintavételezési frekvenciája 48 kHz, ezért az eredményezett csv fájl körülbelül 826 000 sort tartalmazott. A csv fájlon már egyszerű volt Studióban szűrő modulokat alkalmazni. Kipróbált ablakméretek mediánfilterre: 1000, 200, 50.



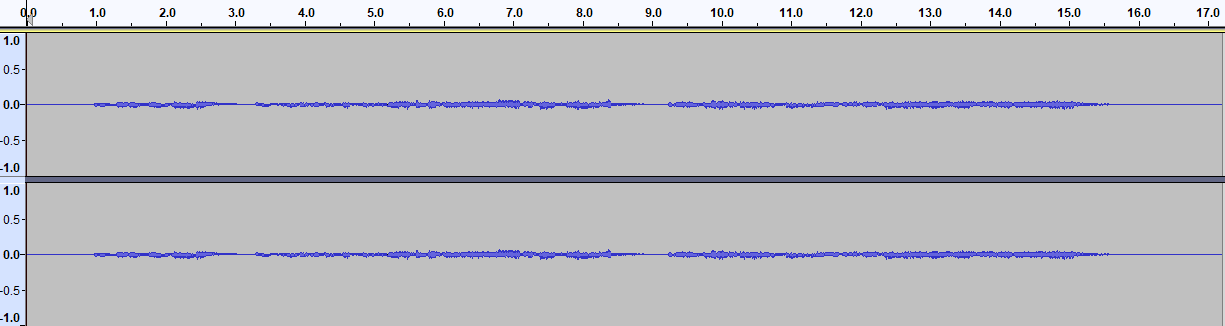
Eredeti hangfájl



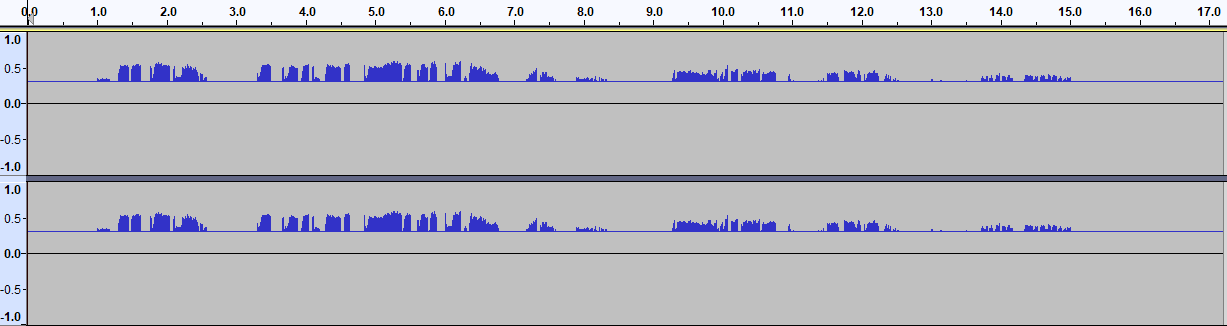
1000 ablakméretű medián filter alkalmazása



200 ablakméretű medián filter alkalmazása

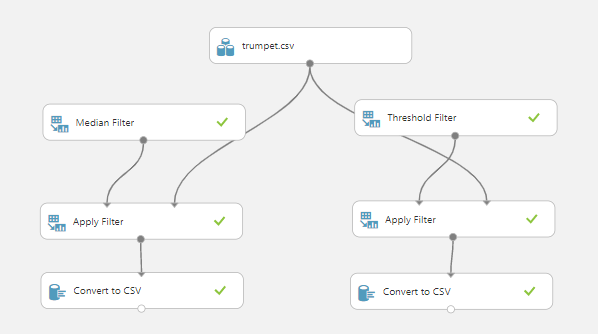


50 ablakméretű medián filter alkalmazása



Küszöbérték filter (10000-nél kisebb intenzitásokat a küszöbértékre állít)

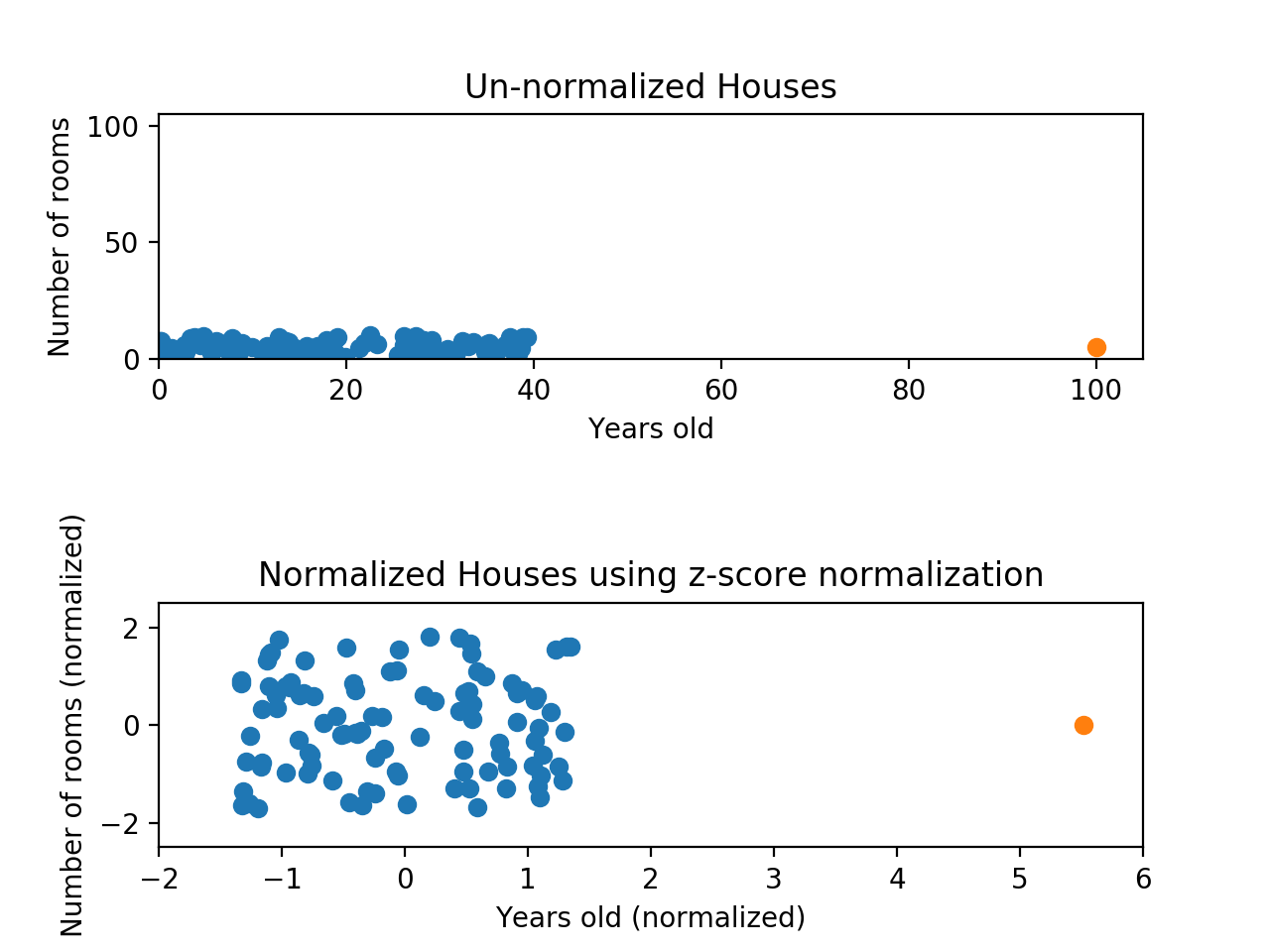
A kísérlet alakja a Studio felületén:



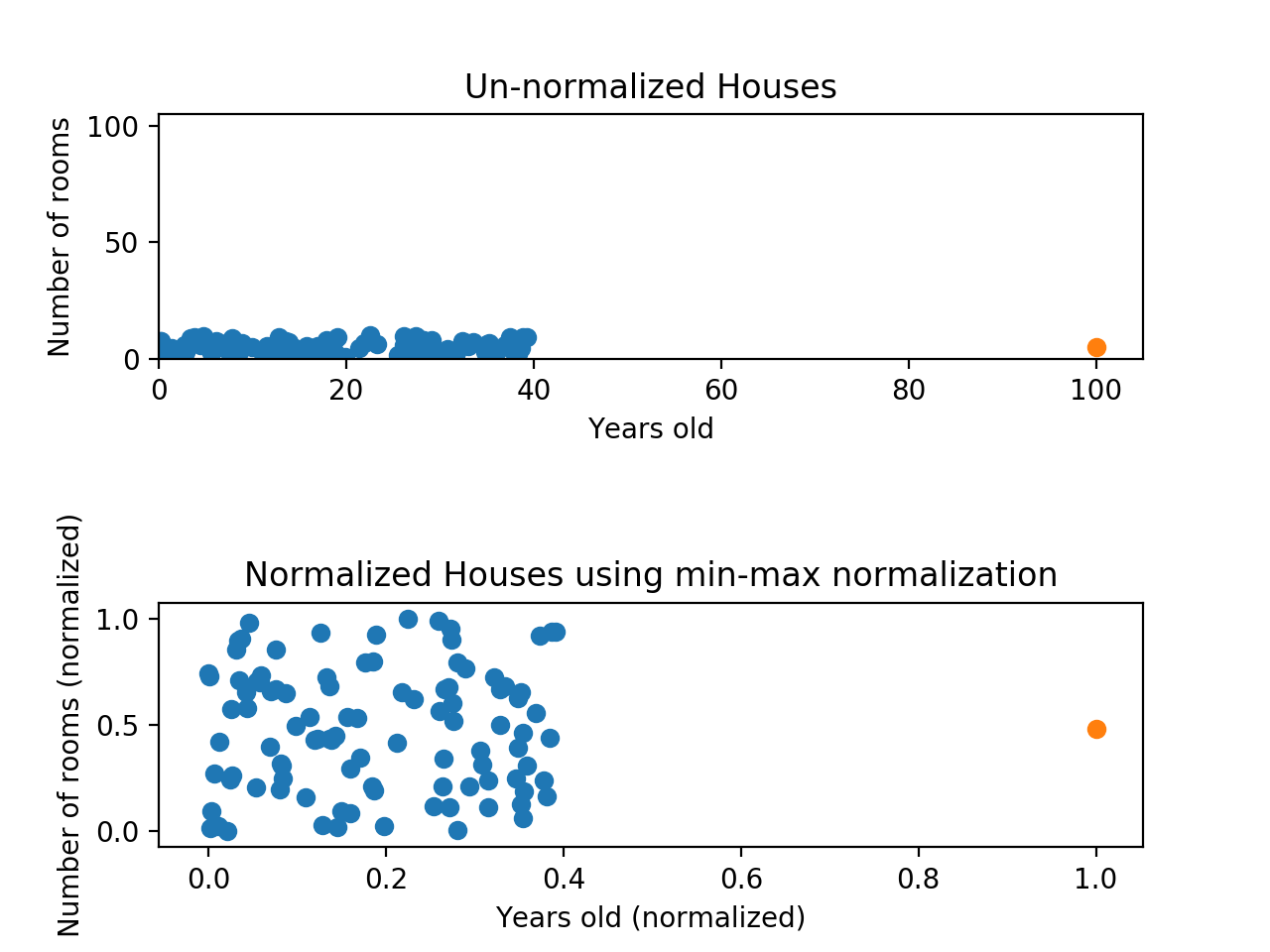
### Normalizációs technikák Azure ML Studio-ban

Azure Machine Learning Studioban az alábbi normalizációs technikák közül lehet választani:

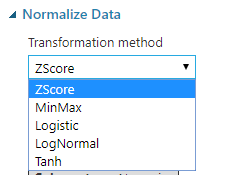
* ZScore: a Z-score eljárás minden értékből kivonja a valószínűségi változó várható értékét, majd elosztja a szórással. Minél közelebb van egy érték a várható értékhez, annál közelebb lesz 0-hoz. Az értékek minden tengelyen ugyanazon intervallumon belül fognak elhelyezkedni, outlierek jelenléte esetén is.



* MinMax: minden értéket a [0,1] intervallumba képez, 0-ának a legkisebb szám fog megfelelni az adatokból, míg 1-nek a legnagyobb. Hátránya, hogy az outlier adatokat rosszul kezeli. Például, ha 99 adatunk 20 és 40 közötti szám, és 1 pedig 100, akkor az összes érték a [0, 1] intervallum elejére lesz beszorítva, míg 1 érték lesz a végén.



* Logistic: az adatokat a szigmoid függvény segítségével a [0, 1] intervallumba képezi.
* LogNormal: a változót log-normális eloszlásra hozza.
* Tanh: az adatok a hiperbolikus tangens függvény segítségével a [-1,1] intervallumba képezi

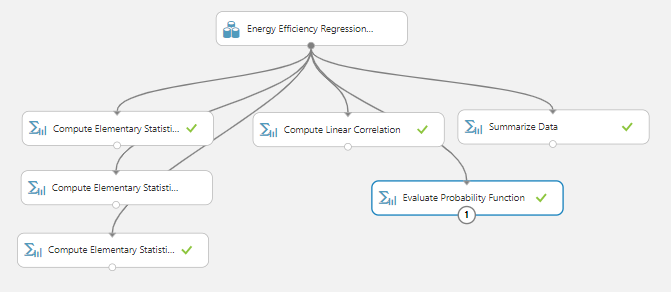


### Statisztikai / Matematikai modulok

Az alábbi statisztikai/matemaikai modulok érhetők el:

* Matematikai művelet alkalmazása (Apply Math Operation)
* Statisztika számolása (Compute Elementary Statistics):
  + Szórásnégyzet
  + Geometriai közép
  + Harmonikus közép
  + k-adik centrális momentum
  + Maximum
  + Minimum
  + Számtani közép
  + Medián
  + Produktum
  + Szumma
  + Skála
  + ...
* Lineáris korreláció számolása (Compute Linear Correlation)
* Eloszlás illesztése (Evaluate Probability Function):
  + Bernoulli
  + Binomiális
  + Egyenletes
  + Gauss
  + Poisson
  + Exponenciális
  + Gamma
  + ...
* Adatok összegző statisztikáinak számolása (Summarize Data)
* Diszkrét értékek cseréje (Replace Discrete Values)
* Hipotézisvizsgálat (Test Hypothesis using t-Test)

Egy egyszerű projektet hoztam létre, amelyben az Energy Efficiency Regression adatszettet vizsgáltam statisztikai modulokkal.



A bal oldali három modul egy-egy statisztikát számolnak ki, ebben az esetben a szórásnégyzet, változó által felvett értékek tartománya és medián értékek.

A lineáris korrelációt számoló egy korrelációs mátrixot ad eredményül.

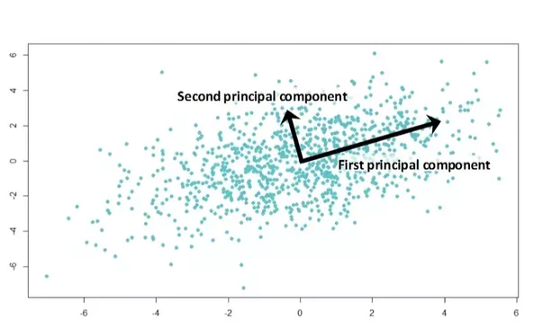
Az adatösszegző modul minden oszlopra megadja az elemek, az egyedi értékek és a hiányzó elemek számát, valamint minimális, maximális és átlag értékeket.

Az eloszlás illesztő ebben az esetben normáleloszlást próbál illeszteni, a paraméterben megadott várható érték és szórás értékekkel. Fontos megjegyezni, hogy minden eloszlás esetében más és már paraméterek megadása szükséges.

### PCA és LDA

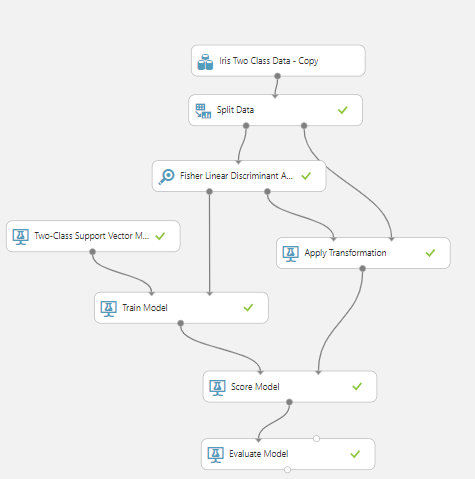
Mindkét módszer dimenziócsökkentési módszer. A PCA (Principal Component Analysis) egy felügyeletlen, míg az LDA (Linear Disciminant Analysis) egy felügyelt módszer, azaz szüksége van osztály címkékre.

A PCA bázistranszformáció által egy olyan bázist keres, mely komponensei mentén az adatok szórása maximális.



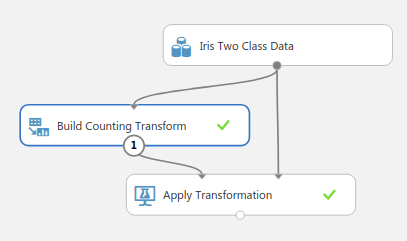
Az LDA olyan altereket keres, melyek mentén az adatokat legkönnyebben szétválaszthatók.

Az alábbi projekt felhasználja az LDA modult, amely segítségével a 4 feature-ből álló klasszikus iris adatszettet 2 feature oszlopra csökkenti. Az LDA modul egyetlen hiperparamétere a dimenzócsökkentés utáni dimenziók száma. Klasszifikálásra a Support Vector Machine módszert megvalósító modult használja. A kísérlet sikeres, a megbecsült osztálycímkék mindegyike helyes.

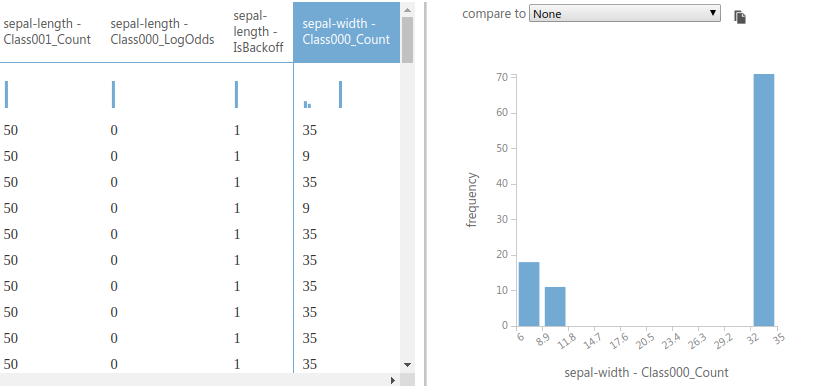


### Counts táblák

A Count táblák az adatszett attribútumain végez megszámlálást, azaz minden lehetséges érték előfordulását megszámolja és összevetést készít a többi értékhez képest.



A Build Counting Tranform eredménye egy transzformációs sablon, amit alkalmazni lehet az adatszettre, megkapva így egy Count táblát.



Ezen kívül, lehetőség van Count táblák exportálására, importálására, összevonására és módosítására.

### További adatmanipulációs modulok

1. Adatok szétválasztása (Tanító / Teszt adatok létrehozása)
2. Oszlop/sor hozzáadás
3. SQL transzformáció végrehajtása
4. Hiányzó adatok kezelése: hiányzó cellák helyébe lehetőség van alapértelmezett értéket beírni, az oszlopban szereplő értékek átlagát vagy mediánját, sor törlése, esetleg teljes oszlop eltávolítása
5. Metaadatok szerkesztése: adattípusok, intervallumok megadása, módosítása, oszlopok kategorikusságának explicit beállítása
6. Duplikált sorok törlése
7. SMOTE (*Synthetic Minority Oversampling Technique*): statisztikai módszer egy alulreprezentált osztályba tartozó adatok számának növelésére.
8. Feature Selection - Jellemző kinyerés
   1. Filter alapú
   2. Fisher Linear Discrimant Analysis

## 

## Tanítási folyamat, modellek

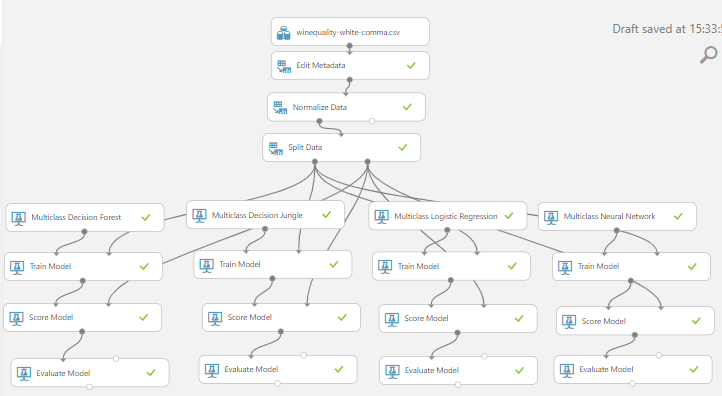
### Gépi tanulás modellek az Azure Machine Learning Studio-ban

1. Bináris klasszifikációra előkészített modellek
   1. Two-class Averaged Perceptron (Átlagolt Perceptron)
   2. Two-class Bayes Point Machine
   3. Two-class Boosted Decision Tree
   4. Two-class Decision Forest (Döntési Erdő)
   5. Two-class Decision Jungle (Döntési Dzsungel)
   6. Two-class Locally Deep Support Vector Machine
   7. Two-class Logistic Regression (Logisztikus Regresszió)
   8. Two-class Neural Network (Neurális Háló)
   9. Two-class Support Vector Machine (Tartóoszlop-vektor automata)
2. Többosztályos klasszifikációra előkészített modellek
   1. Multiclass Decision Forest (Döntési Erdő)
   2. Multiclass Decision Jungle (Döntési Dzsungel)
   3. Multiclass Logistic Regression (Logisztikus Regresszió)
   4. Multiclass Neural Network (Neurális Háló)
3. Regresszióra előkészített modellek
   1. Bayesian Linear Regression (Bayes lineáris regresszió)
   2. Boosted Decision Tree Regression
   3. Decision Forest Regression (döntési erdő regresszió)
   4. Fast Forest Quantile Regression
   5. Linear Regression (lineáris regresszió)
   6. Neural Network Regression
   7. Ordinal Regression
   8. Poisson Regression
4. Klaszterizáló modellek
   1. K-means (K-közép)

### Többosztályos klasszifikátor modellek összehasonlítása I.

Az experiment keretein belül a Studióban levő 4 többosztályos klasszifikátor teljesítményét hasonlítottam össze az UCI Machine Learning Repository-n található Wine Quality adatbázison. Az adatokat normalizáltam a Z-score módszerrel. Előfeldolgozási lépésként, opcionálisan, a Linear Discriminant Analysis és Principal Component Analysis módszereket is kipróbáltam.

A projekt:



Modellek kiértékelése a teszthalmazon (pontosság százalékban):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Döntési erdő | Döntési dzsungel | Többosztályos logisztikus regresszió | Neurális háló |
| Nincs előfeldolgozás | 66.42% | 63.77% | 53.97% | 56.73% |
| LDA (8 dim.) | 64.08% | 59.48% | 52.24% | 56.12% |
| LDA (9 dim.) | 63.87% | 62.34% | 52.49% | 55.40% |
| PCA (9 dim.) | 66.53% | 61.53% | 52.55% | 56.63% |
| PCA (5 dim) | 62.14% | 61.42% | 50% | 54.08% |

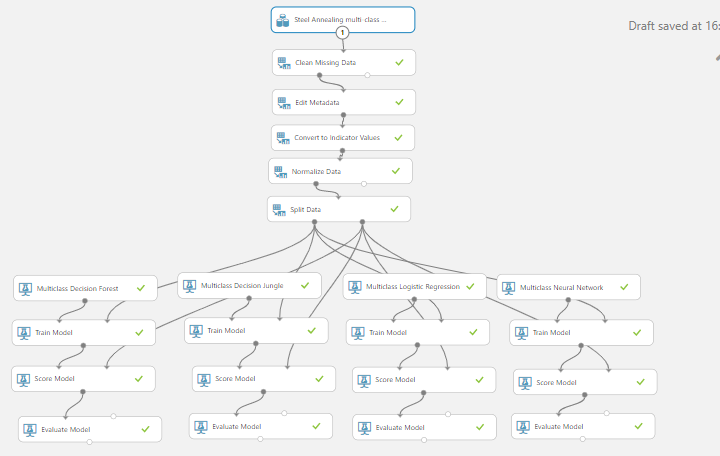
Látható, hogy ez az adatszett nem triviálisan klasszifikálható, és a Studió módszerei nem képesek kapcsolatot teremteni az attríbutomok és a címkehalmaz között.

### Többosztályos klasszifikátor modellek összehasonlítása II.

Az experiment keretein belül a Studióban levő 4 többosztályos klasszifikátor teljesítményét hasonlítottam össze az Azure mintaadatszett gyűjteményében található Steel Annealing adatbázison. A Clean Missing Data modul a hiányzó értékeket pótolja, zérusokkal. Mivel az oszlopok többsége kategorikus változó, ezért a one-hot elkódolást végző Convert to Indicator Values modult használtam fel. Az adatokat normalizáltam a Z-score módszerrel.

Az adathalmaz a 798 sort és 85 attríbutum oszlopot tartalmaz közvetlenül a klasszifikátor előtti lépésben.

A projekt:



Mért pontosságok a teszthalmazon:

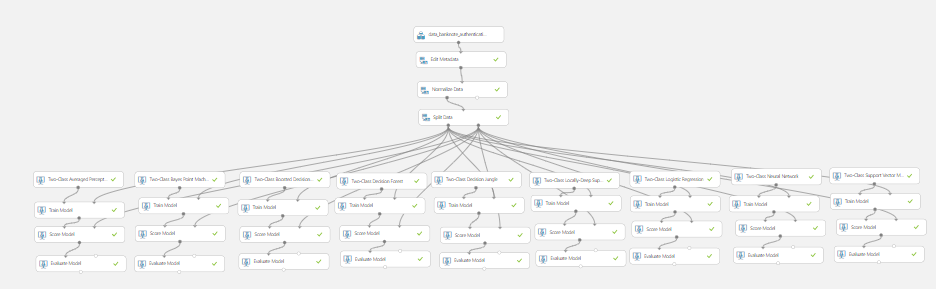
1. Döntési erdő: 99.4%
2. Döntési dzsungel: 99.4%
3. Többosztályos logisztikus regresszió: 97.9%
4. Neurális háló: 94.9%

Ebben az esetben nyilvánvaló, hogy a mind a négy modell megállja a helyét és helyesen feltárja az oszlopok és a osztálycímke között levő rejtett összefüggéseket.

### Bináris klasszifikátor modellek összehasonlítása

A kísérlet a Studióban levő összes bináris klasszifikátor teljesítményét hasonlítja össze. Az adatszett a UCI Machine Learning repositoryn található Banknote Authentication Data Set. Az adatokat normalizáltam a Z-score módszerrel.

A kilenc párhuzamosan futó regressziós modell felülnézetből:



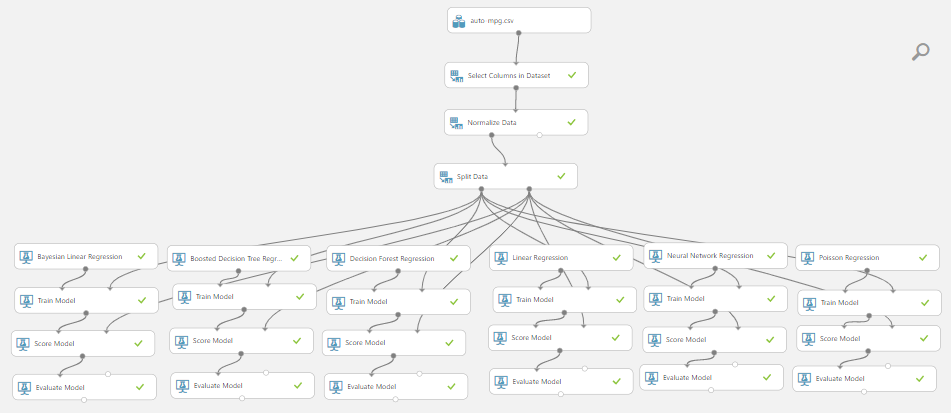
Mért pontosságok a teszthalmazon:

1. Two-class Averaged Perceptron (Átlagolt Perceptron): 98.8%
2. Two-class Bayes Point Machine: 99.1%
3. Two-class Boosted Decision Tree: 99.4%
4. Two-class Decision Forest (Döntési Erdő): 99.4%
5. Two-class Decision Jungle (Döntési Dzsungel): 99.1%
6. Two-class Locally Deep Support Vector Machine: 100%
7. Two-class Logistic Regression (Logisztikus Regresszió): 97.1%
8. Two-class Neural Network (Neurális Háló): 99.7%
9. Two-class Support Vector Machine (Tartóoszlop-vektor automata): 97.7%

### Regressziós modellek összehasonlítása I.

A kísérlet a Studióban levő regresszor modellek teljesítményét hasonlítja össze. Az adatszett a UCI Machine Learning repositoryn található Auto MPG Data Set, amelyben különböző autók fogyasztása van becsülve a jármű paraméterei alapján. Az adatokat normalizáltam a Min-max módszerrel.

A hat párhuzamosan futó regressziós modell felülnézetből:



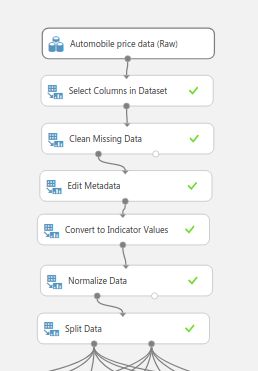
Predikciós hibák (RMSE - átlagos négyzetes hiba):

1. Bayesian Linear Regression: 0.126
2. Boosted Decision Tree Regression: 0.077
3. Decision Forest Regression: 0.084
4. Linear Regression: 0.093
5. Neural Network Regression: 0.102
6. Poisson Regression: 0.086

### Regressziós modellek összehasonlítása II.

A kísérlet a Studióban levő regresszor modellek teljesítményét hasonlítja össze. Az adatszett a Studio egyik minta-adatszettje, az Automobile price data, amely különböző autók paraméterei alapján határozza meg az autó árát.

Az előfeldolgozási lépések grafikus formája:



A Select Columns in Dataset modul segítségével elhagyunk két oszlopot, amelyek nem szolgálnak hasznos információkkal a modell tanítási folyamatában. Mivel van két oszlop, amely néhány sorra nem tartalmaz adatot, ezért a Clean Missing Data modult felhasználva a hiányos cellákba a program beírja az oszlop értékeinek átlagos értékét.

Az Edit Metadata modul a kategórikus oszlopokra explicit beállítja az adattípusukat és a változó fajtáját. A Convert to Indicator Values one-hot enkódolást végez a kategórikus oszlopokon. A Normalize Data normalizálja a numerikus oszlopokat Min-Max normalizációval.

A projekt többi része megegyezik az előző regressziós kísérlettel, ugyanaz a 6 modell lett kipróbálva.

Predikciós hibák (MAE - átlagos abszolút hiba):

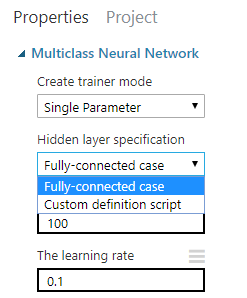
1. Bayesian Linear Regression: 1943.5 dollár
2. Boosted Decision Tree Regression: 1964 .6 dollár
3. Decision Forest Regression: 2727.9 dollár
4. Linear Regression: 2032.5 dollár
5. Neural Network Regression: 6436.3 dollár
6. Poisson Regression: 3018.0 dollár

A neurális háló magas hibája furcsa jelenség, a prediktált értékek vizsgálata után kiderült, hogy a modell majdnem ugyanazokat az értékeket becsüli. A jelenség oka, hogy a becsülendő oszlop (autó ára) nagyságrendje túl nagy, és instabil gradiens problémát okoz a neurális hálóban, így az nem tanul. Erre megoldás a prediktálandó oszlop normalizálása.

### Net# nyelv - Személyre szabott neurális hálók

Neurális háló modulok esetében a fully-connected architektúra alternatívája egy saját kialakítású architektúra, melyet egy szkript segítségével lehet megírni. A szkript nyelve a Net#, egy speciális Azure nyelv, mely neurális háló architektúra specifikálára van tervezve.

Bármely neurális modul behúzása után, a tulajdonságok panelban, a Hidden layer specification tulajdonságon belül a Custom definition script kiválasztása után létrejön a saját szkript, amellyel tetszőleges architekúrát ki lehet alakítani.



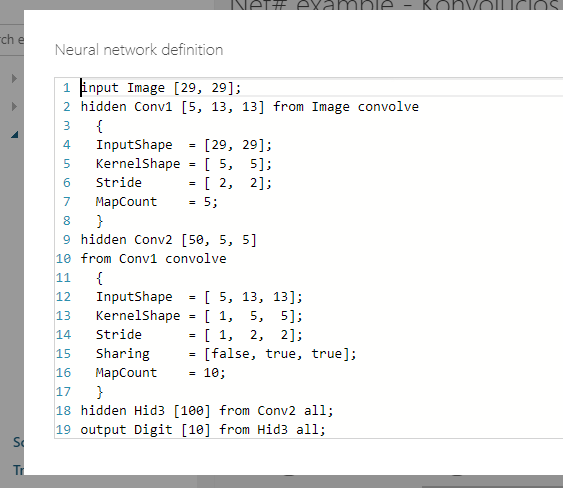
A Net# nyelv grammatikája és szabályrendszere nagyban hasonlít a C/C++ -ra: érzékeny a kis és nagy betűkre, megjegyzéseket dupla ferde vonallal kell indítani, konstansok deklarációja hasonló. A kód főleg a különböző rétegek deklarálásából tevődik össze, és azok paramétereinek megadásával. Különböző típusú rétegek léteznek: teljesen összekötött (fully connected), konvolúciós, pooling. Minden rétegnek nevet kell adni. Közbülső rétegek deklarációjánál specifikálni kell az előző réteget, melyet a nevén keresztül lehet.

### Konvolúciós háló Net# nyelv segítségével

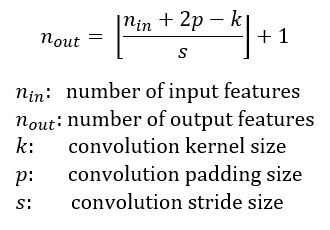
A Net# szkriptek lehetővé teszik konvolúciós hálók programozását.

A folyamat első lépése egy képes adattár feltöltése, mint például a Caltech 101 adatbázis. ([link](http://www.vision.caltech.edu/Image_Datasets/Caltech101/)) Azure ML Studióban erre több lehetőség is van:

* Az Import Images modul alkalmazása (OpenCV Library Modules). Ennek segítségével nyilvános URL-en és Azure Blob storage-ban eltárolt adatszetteket lehet betölteni a projektbe.
* CSV vagy TSV fájl feltöltése, melyekbe bele vannak kódolva a képek.
* adatok feltöltése ZIP fájlba tömörítve, majd egy belső Python vagy R szkript segítségével az adatok manipulálása és előkészítése a modell számára. Nagy méretű adatbázisok esetén ez a módszer hátrányos lehet időigénye miatt.



A rétegek specifikációjánál ügyelni kell arra, hogy a réteg kimeneti méretei egyezzenek a beálított kernel méret, stride, padding és map count tulajdonságok által meghatározott méretekkel. Az alábbi képlet segítségével a konvolúciós réteg kimenete könnyen számítható:



Készítettem egy kísérletet, melyben egy Net#-ban megírt konvolúciós hálót tanítottam be az MNIST adatszettben szereplő számjegyek klasszifikálására. A háló struktúrája az alábbi: bemeneti réteg, két konvolúciós réteg, 100 méretű sűrű réteg és kimeneti réteg. A képeket ZIP formában töltöttem fel, majd egy saját szkript segítségével a képes adatszettet egy DataFrame objektummá alakítottam. Ebben az esetben a mappanévből kinyerhető a ketegórianév, illetve a kép lekódolható DataFrame objektumba úgy, hogy minden pixel intenzitásértéke egy külön attribútum, azaz egy 30\*30 -as kép esetén 900 új attribútumot kell létrehozni. A modell 97%-os pontosságot ér el. A tesztadatokra való kiértékelés konfúziós mátrixa:

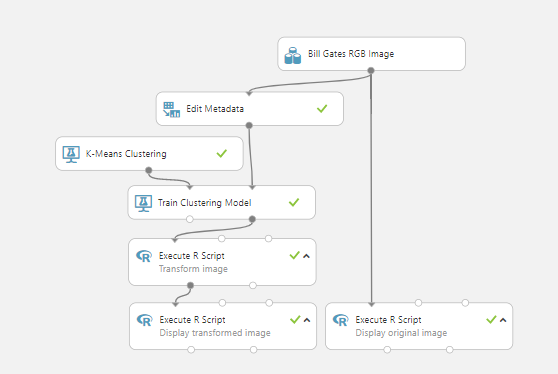


Linkek: [link1](https://docs.microsoft.com/en-us/azure/machine-learning/studio/azure-ml-netsharp-reference-guide), [link2](https://blogs.technet.microsoft.com/machinelearning/2015/02/16/neural-nets-in-azure-ml-introduction-to-net/) .

### Klaszterizálás - Színes kép színkvantálása

A projekt a K-közép klaszterizáló algoritmus működését példázza. Egy színes kép pixeleit két klaszterbe soroljuk, majd a “binarizált” képet megjelenítjük. A bemeneti kép egy csv fájl, amely soronként egy-egy pixel vesszővel elválasztott X, Y, R, G, B értékeit tartalmazza.

A projekt grafikus reprezentációja:



A K-means modul paramétereiben fontos megadni a centroid-számot, ami ez esetben 2. Továbbá, lehetőség van inicializációs algoritmus, metrika és iterációszám paraméterek megadására.

A legalsó szinten levő két R szkript megjelenítik az eredeti képet, valamint a kvantált képet.



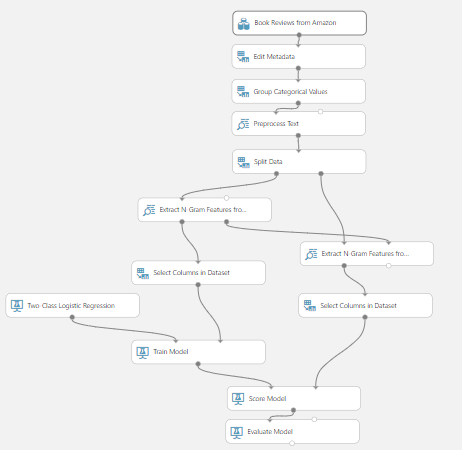
### Szöveganalitika, érzelmi analízis könyv-véleményeken

A kísérlet célja egy olyan modell létrehozása, amely szöveges véleményeket feldolgozva prediktálja, hogy a vélemény a negatívan vagy a pozítivan értékelt visszajelzések kategóriájába tartozik. A projekt nagy mértékben felhasználja a Studio szöveganalitikai moduljait.

Az Azure ML Studióban az alábbi modulok érthetők el a Text Analytics fül alatt:

* Detect Languages (nyelv felismerése)
* Extract Key Phrases from Text (kulcsszavak kinyerése)
* Preprocess Text (szöveg előfeldolgozása: kisbetűvé alakítás, számok törlése, stb...)
* Named Entity Recognition (szövegben szereplő entitások megnevezése)
* feature kinyerési modulok

A kísérlet grafikus formája:

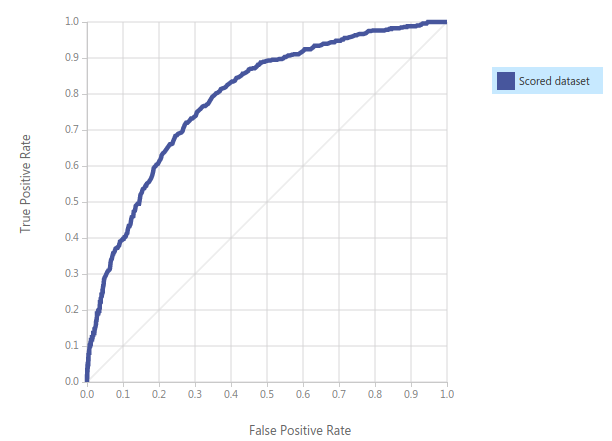


Az adatbázis egy Amazonon árult könyről írt vélemények és értékelések gyűjteménye. Az első oszlop egy 1-2 és 4-5 közötti értékelés, a második oszlop egy szövegblokk.

A Group Categorical Values modul az értékeléseket negatív (1-2), valamint pozitív (4-5) kategóriába csoportosítja. A Preprocess Text modul törli a számokat, jeleket, e-mail címeket, URL-eket; lemmatizálást végez, valamint a betűket mind kisbetűvé állítja.

Ezután, numerikus featureöket nyerünk ki a szövegekből az Extract N-Gram Features from Text modullal, hiszen a gépi tanulás modell számértékekkel dolgozik. A kapott adatok rendkívül nagy dimenzionalitásúak lesznek, hiszen annyi oszlop származik, ahány egyedi szó van, mivel az eljárás a one-hot encoding módszerhez hasonlít.

A következő lépés egy bináris klasszifikátor betanítása ezeken az adatokon, ebben az esetben egy többváltozós logisztikus regresszió. Utolsó lépésben a modell kiértékelése következik a tesztadatokon. A modell teljesítményének kiértékelésére használt az ROC görbe, ami a helyes pozitívok és a fals pozitívok arányát fejezi ki:

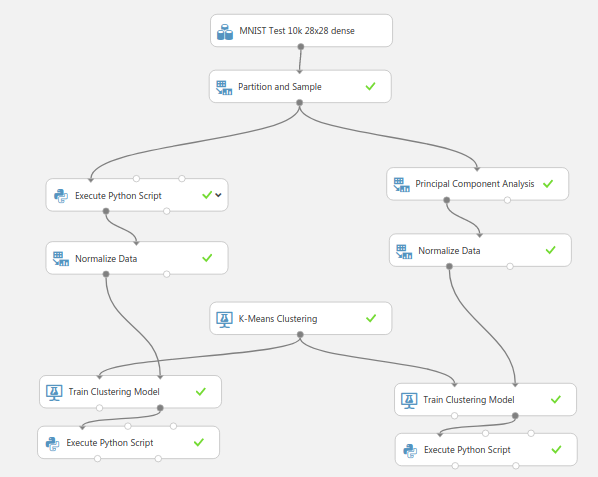


A klasszifikátor pontossága 80,8%.

### PCA és t-SNE dimenziócsökkentés, majd K-közép klaszterizálás

A kísérleten belül klaszterizálni próbáltam az MNIST adatszettben levő 28x28 méretű számjegy képeket.

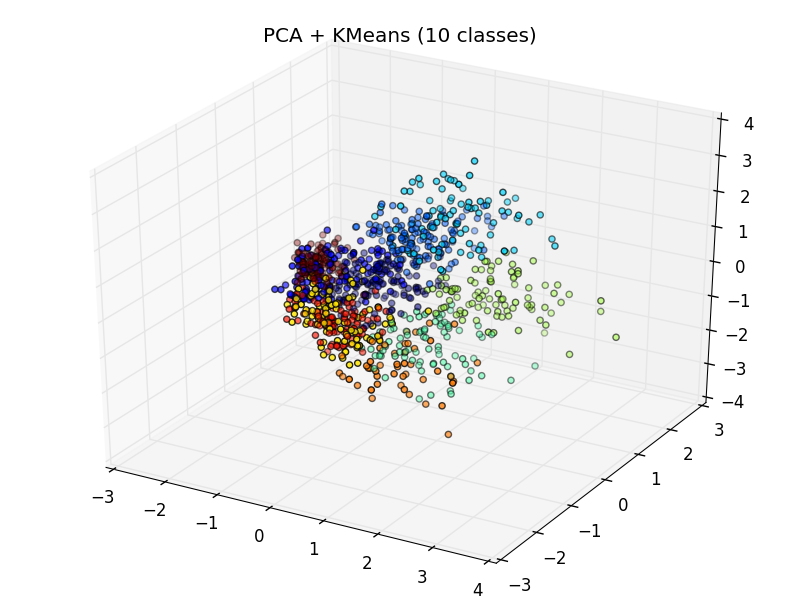
A projekt grafikus formája:



Mivel az adatok dimenzionalitása 28x28, ezért a két vizsgált dimenziócsökkentő módszert használva az adatokat 3 dimenziójura hozom, könnyű vizualizálás érdekében.

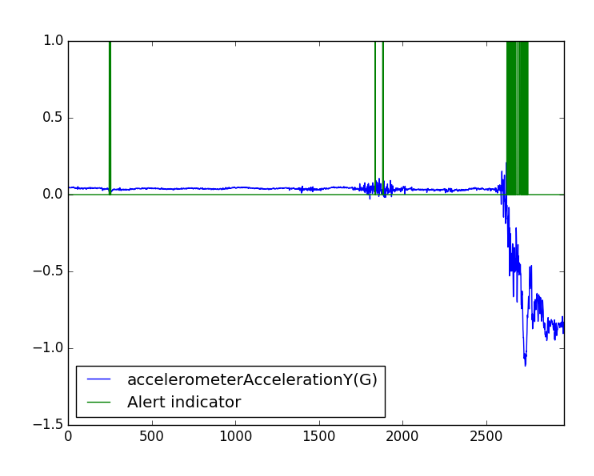
A K-közép klaszterizáló centroidszám hiperparamétere 10 ebben az esetben, mivel 10 számjegy van.

PCA módszer eredménye:



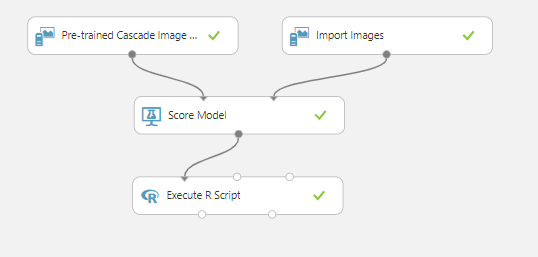
### Anomália Detekció Idősoron

Azure Machine Learning Studioban elérhető egy idősoron való anomáliadetekciós modul, melyet egy külön kísérleten keresztül próbáltam ki. Az adatok egy gyorsulásmérő által rögzített értékek. Az algoritmus egy ablakkal vonul végig az adatokon, és kimagasló értékeket vagy meredek negatív/pozitív trendeket anomáliának nyilvánít. Állítható paraméterek az ablak mérete; a küszöbérték, melyen felüli anomáliapontszám anomáliának minősül; a vizsgált változás fajta (kiugró érték vagy szokatlan trend). A modul eredménye egy olyan DataFrame, ami minden adatelemre tartalmazza az anomália pontszámot és egy bináris számot, amely megadja, hogy az adott adatpont anomália vagy sem.



### Arcfelismerés

A beépített, előre betanított arcklasszifikátort alkalmazva (Cascade Image Classification), lehetőség van feltöltött képeken arcminták felismerésére. Az Import Images modul segítségével az Azure tároló területről lehet a képeket beolvasni. Ehhez egy Azure Storage fiók szükséges, amely Storage (general purpose v1) típusú, és ezen belül, a képek egy blob/container típusú tárolóba vannak elhelyezve. A fájlokhoz való hozzáféréshez a fiók nevét, a container elérési útját és a hozzáférési kulcsot kell megadni. A képek felbontására felső korlát van előírva, a kép maximum 409.000 pixelt tartalmazhat.



A modell kimenete egy valószínűség érték, ami az arcminta képen való jelenlétének a valószínűségét jelzi.

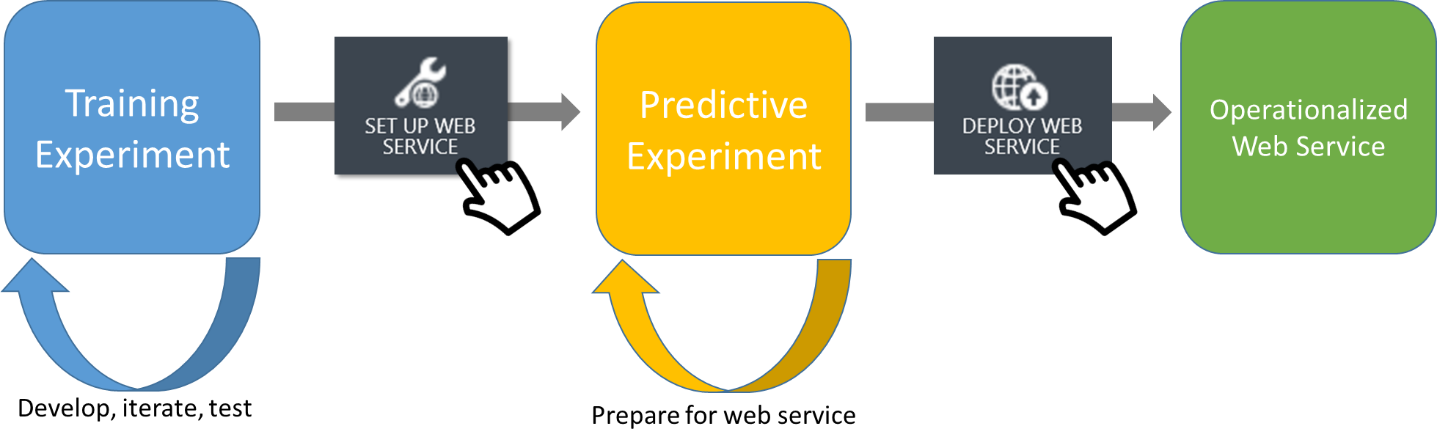
### További modulok

* Előre betanított arc felismerő / klasszifikáló modul
* Python script futtatása
* R script futtatása

## Üzembe helyezés, külső felhasználás

### Experiment deployolása, mint Webszolgálatás

A bekonfigurált folyamatot, modellt igény esetén exportálni lehet, mint web-szolgáltatás. A felület alján, a  *Set Up Web Service* gommbal először felkonfigurálni lehet a szolgáltatást, amely egy új tabban fog megnyilni a Studio felületén. A szolgáltatást ezen a ponton publikálni/deployolni lehet, és API-ként egy nyilvános url-en elérhető lesz, amelyhez egy API-kulcsot is kapunk.

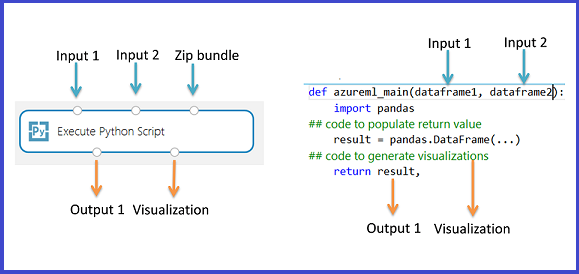


A felállított webszolgáltatás a Studio *Web Services* menüjében lesz látható, ahol azt ki lehet próbálni, illetve információt kapni a használatáról. A kérésben megadni szükséges paraméterek, adatok; a válasz tartalma, strukturája, valamint mintakódok szerepelnek ezen információk között.

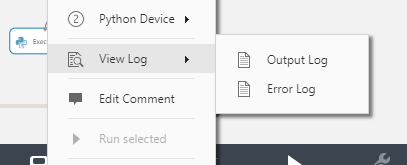
## Egyéb funkciók

### Saját Python/R szkript futtatása

A behúzott adatbázisok automatikusan pandas dataframe-vé konvertálódnak. Tetszőleges adatokat, python könyvtárakat a Zip bundle paraméter segítségével lehet feltölteni, amelyet harmadik bemenetként workspacre feltöltött zip fájlok közül kell kiválasztani.



A szkript kimenete két naplófájlba rögzül, a standard kimenet az “Output Log”-ba, míg a hibaüzenetek az “Error Log”-ba. Mindkettő a modulra való jobb kattintással,majd a “View Log” menüpont megnyitásával érhető el.



Lehetőség van adatvizualizálásra a matplotlib könyvtár segítségével. Az elkészült diagram a “Python device” fül alatt, a “Visualize” gombra való kattintással érhető el.

### Notebook funkcionalitás az Azure ML Studio-n belül

Lehetőség van meglévő és új notebookok futtatására, szerkesztése és létrehozására.

Az elérhető könyvtárak száma nem olyan magas, mint a Google Colab esetében, és nincs lehetőség új könyvtárak telepítésére.

Létezik dedikált Azure notebook szolgáltatás (<https://notebooks.azure.com/>), mely már lehetőve teszi a könyvtárak telepítését.

A kutatást az „Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein” (EFOP-3.6.3-VEKOP-16-2017-00002) című projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.