Liceul Teoretic “Grigore Moisil” Tulcea

**APLICATII MACHINE LEARNING**

LUCRARE PENTRU ATESTAREA COMPETENTELOR PROFESIONALE

Coordonator: Parlitu Rodica

Elev: Saraev Stefan

Clasa: a XII-a B

2019

TEMA

Sa se realizeze proiectul cu tema ….

Se va folosi limbajul JavaScript.

CUPRINS

[1. DESPRE LIMBAJ 4](#_Toc377280)

[2. ALCATUIREA LUCRARII 5](#_Toc377281)

[2.1 SCHEMA LOGICA 5](#_Toc377282)

[2.2 DESCRIEREA LUCRARII 6](#_Toc377283)

[3. MANUAL DE UTILIZARE 12](#_Toc377284)

[Optiunea 1 13](#_Toc377285)

[Optiunea 2 15](#_Toc377286)

[Optiunea 3 17](#_Toc377287)

[4. BIBLIOGRAFIE 18](#_Toc377288)

[5. ANEXE 19](#_Toc377289)

# DESPRE LIMBAJ

JavaScript (JS) este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea siturilor web, dar este folosit și pentru accesul la obiecte încastrate (embedded objects) în alte aplicații. A fost dezvoltat inițial de către Brendan Eich de la Netscape Communications Corporation sub numele de Mocha, apoi LiveScript, și denumit în final JavaScript.

În ciuda numelui și a unor similarități în sintaxă, între JavaScript și limbajul Java nu există nicio legătură. Ca și Java, JavaScript are o sintaxă apropiată de cea a limbajului C, dar are mai multe în comun cu limbajul Self decât cu Java.

Până la începutul lui 2005, ultima versiune existentă a fost JavaScript 1.5, care corespunde cu Ediția a 3-a a ECMA-262, ECMAScript, cu alte cuvinte, o ediție standardizată de JavaScript. Versiunile de Mozilla începând cu 1.8 Beta 1 au avut suport pentru E4X, care este o extensie a limbajului care are de a face cu XML, definit în standardul ECMA-357. Versiunea curentă de Mozilla, 1.8.1 (pe care sunt construite Firefox și Thunderbird versiunile 2.0) suportă JavaScript versiunea 1.7.

Am ales sa scriu acest atestat in JavaScript, si nu in C, C++ sau Java, deoarece consier ca este mai avantajos pentru utilizator sa dispuna de unelte de lucru, pe orice tip de platforma (Linux, MacOS, Windows), avandu-le la dispozitie prin intermediul internetului si, implicit, a paginilor WEB.

Un avantaj al scrierii acestui program in JavaScript este ca ofera, prin intermediul consolei, feedback in timp real, lucru ce usureaza munca utilizatorului. Acesta poate observa foarte usor eventualele erori care pot aparea datorita utilizarii incorecte a programului.

Un alt avantaj este capacitatea de implementare a proiectului intr-o pagina WEB, acesta nefiind altceva decat 10 fisiere ‘.js’. Acest lucru inseamna ca fiecare institutie, in functie de propriile nevoi in cercetare, poate prelua si adapta acest proiect spre a realiza diferite aplicatii.

# 2. ALCATUIREA LUCRARII

## 2.1 SCHEMA LOGICA

ChoseWhatToDo

Scrie

rezultate.txt

Prelucrare date

Citeste

\*.nn.txt, variabile.txt

Afiseaza

imagine

Prelucrare date

Citeste

Click-uri pe imagine

Citeste

\*.png/\*.jpg

Citeste

hidden

Scrie

\*.nn.txt

Prelucrare date

Citeste

variabile.txt, rezultate.txt, config.txt

Citeste

Inputs,hidden,outputs

Utilizeaza o ecuatie

Determina linia orizontului

Determina o ecuatie

## 2.2 DESCRIEREA LUCRARII

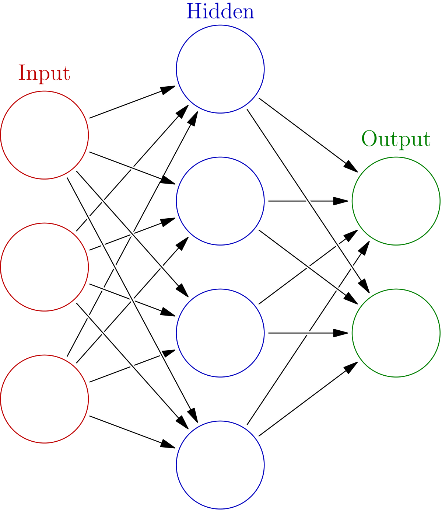
Ce este Machine Learning? Unii il definesc ca inceputul unui AI (Inteligenta Artificiala). Eu il consider un algoritm magnific prin care noi, ca oameni, am reusit sa scriem pe hartie cum gandim. Algoritmul Machine Learning este capabil sa clasifice seturi de date dupa caracteristici (pe care le deduce singur), sa deduca ecuatii si sa realizeze predictii precise cu privire la situatii noi, cu care nu s-a mai intalnit. Daca va amintiti, pe Facebook, exista postari prin care, fiind date o lege de compozitie \* si 3 rezultate ale (x\*y), li se cere cititorilor sa deduca legea de compozitie \*. Ideea din spatele acestui exemplu este aceea ca, pe masura ce ne adancim in abstract, avem nevoie de programe capabile sa faca asta pentru noi: sa deduca ecuatii. Noi putem gasi raspunsul doar pentru un numar limitat de seturi de variabile de intrare, iar aceasta o facem intr-un mod experimental. Acest fapt inseamna mult timp alocat cercetarii si multe resurse folosite, iar ca sa gasim ecuatia, avem nevoie fie de “inspiratie divina”, fie de foarte multe experimente, care inseamna si mai multe resurse. In schimb, daca am folosi algoritmul Machine Learning pentru a determina ecuatia, am reusi sa avem ceva concret in mai putin de 2 zile, fara prea mari batai de cap.

De ce spun ca este foarte folositoare? In viata de zi cu zi, noi observam, clasificam si catalogam lucruri, deducem situatii si rezultate, folosim numere. Daca exista 3 lucruri pentru care calculatoareale chiar ne sunt folositoare, acelea sunt ca lucreaza foarte repede cu numere, au capacitate de stocare foarte mare si pot face acelasi lucru de foarte multe ori, fara sa se plictiseasca. Ce inseamna asta? Oriunde exista nevoie de a se opera cu numere si de a stoca multa informatie, calculatoarele sunt mai bune decat oamenii. Un lucru nou, pentru oameni, este ideea de a transforma lucrurile din jur in numere. Astfel, putem scrie noi algoritmi care sa lucreze cu aceleasi numere, doar semnificatia lor fiind alta. Cum am spus mai sus, Machine Learning poate clasifica si deduce chestii. Astfel, oriunde exista nevoie de clasificare, de deducere, acolo se poate introduce algoritmul acesta. Un exemplu mai concludent este acesta: pentru a se detecta o anumita boala la ochi, doctorul nu poate decat sa se uite la o poza si sa isi dea cu parerea daca pacientul este sanatos sau nu. Insa aceasta parere este influentata de lumina din camera, de atentia doctorului, de cunostintele acestuia de pana in acel moment. Astfel, pentru ca verdictul de sanatos este dat doar de interpretarea unei poze, de ce sa nu lasam aceasta interpretare pe seama unui calculator? Asta au facut baietii de la Google si se pare ca precizia cu care se detecteaza persoane bolnave a crescut de la 40% la 60%. Poate nu vi se pare mult, insa, cat timp este vorba despre viata oamenilor, orice numar in plus conteaza.

Proiectul meu presupune o mica prezentare grafica a unei aplicatii a algoritmului de Machine Learning si o metoda simpla de a crea si utiliza o Retea Neuronala capabila de a aproxima o ecuatie sau de a realiza o clasificare.

Voi prezenta secventele de baza din cod, ca dupa aceea sa le unesc si sa explic cum functioneaza acest proiect.

Ce este o Retea Neuronala? Eu ii spun creier, pentru ca asa este construit. Are la baza mai multe elemente, numite Neuroni, ce preiau mai multe date de intrare, le prelucreaza si genereaza date de iesire care sunt folosite mai departe ca date de intrare. In cod, am definit un obiect numit ‘Neural Network’, ce are mai multi parametrii, precum ‘inputs’, ‘hidden’, ‘outputs’, ‘lr’, ‘trainings’, ‘m1’, ‘m2’, ‘bh’, ‘bo’, si are mai multe functii: ‘estimate’, ‘evolve’. O retea neuronala are aceasta forma:



In aceasta imagine, neuronii rosii reprezinta variabilele de intrare, cei albastrii sunt neuroni cu scop de calcul, iar cei verzi reprezinta variabile de iesire. In cod, ‘inputs’, ‘hidden’ si ‘outputs’ reprezinta numarul de neurni de pe stratul ‘Input’, ‘Hidden’, respectiv ‘Output’. Variabila ‘trainings’ reprezinta numarul de apelari al functiei ‘evolve’. Variabilele ‘m1’, ‘m2’, ‘bh’, ‘bo’ sunt obiecte de tip ‘Matrix’, adica matricile din matematica. Astfel, ‘m1’ reprezinta matricea asociata tuturor conexiunilor dintre stratul ‘Input’ si stratul ‘Hidden’, iar ‘m2’ reprezinta matricea asociata tuturor conexiunilor dintre stratul ‘Hidden’ si stratul ‘Output’. ‘bh’ si ‘bo’ sunt niste matrici asociate unor legaturi ascunse, ce au rolul de a asigura ca datele de iesire sunt diferite de 0, chiar daca toate datele de intrare sunt egale cu 0.

Functia ‘estimate’ preia parametrul ‘input’, ce reprezinta datele de intrare si care este de tip Array, si returneaza un alt Array, salvat in variabila ‘out’, ce reprezinta datele de iesire. Astfel, sunt create 2 matrici, ‘in\_m’, ce contine datele de intrare, si ‘out\_m’, ce va contine datele de iesire. Apoi, este creata matricea ‘hidden’, care este suma dintre matricea ‘bh’ produsul matricilor ‘in\_m’ si ‘m1’. Apoi, fiecare element din matricea ‘hidden’ este supus functiei ‘sigmoid’, ce reprezinta o functie de activare: in urma acesteia vor rezulta numere intre 0 si 1. Functia ‘sigmoid’ este aceasta:

Apoi, matricea ‘out\_m’ ia valoarea sumei dintre ‘bo’ si produsul dintre ‘hidden’ si ‘m2’, dupa care elementele acesteia sunt din nou supuse functiei de activare ‘sigmoid’. Sunt preluate elementele din matricea ‘out\_m’ si salvate in variabila ‘out’, care este returnata.

Functia ‘evolve’ este scrisa pe baza algoritmului ‘Gradient Descent’, ce presupune ajustarea valorilor din matricile ‘m1’, ‘m2’, ‘bh’, ‘bo’. Pe scurt, algoritmul calculeaza eroarea dintre rezultatul cautat si cel calculat si o repartizeaza pe fiecare valoare din cele 4 matrici mentionate mai sus. Daca am adauga la fiecare valoare eroarea proprie, Reteaua Neuronala ar oferi raspunsul perfect pentru cazul cautat, dar nu si pentru celelalte situatii. Aici intervine variabila ‘lr’, ce reprezinta Procentul de invatare. Erorile calculate sunt inmultite cu acest procent inainte de a fi adaugate valorilor din matrici. De exemplu, noi nu dorim ca reteaua sa se adapteze astfel incat sa ne ofere doar rezultatul adunarii dintre 1 si 4, deoarece noua ne trebuie sa putem afla suma dintre 1 si 2, si de aceea nu ii permitem sa se modifice in totalitate, ci doar partial, ca sa se poata indrepta spre aproximatia dorita. Daca procentul este prea mic, erorile devin minuscule, iar algoritmul va necesita foarte mult timp pentru a aproxima ceva. Daca procentul este prea mare, atunci ajustarile se vor anula reciproc, caz in care nu se va putea aproxima niciodata ecuatia cautata.

In fisierul ‘NeuralNetwork.js’, pe langa obiectul ‘Neural Network’, am mai definit un obiect, ‘Matrix’. In acesta voi stoca matrici de numere cu care voi putea face urmatoarele operatii: calculul transpusei, adunarea matricilor, produsul matricilor.

In fisierul ‘Interface.js’ am definit functia ‘randomize’ ce are rolul de a amesteca valorile dintr-un Array.

In fisierul ‘ImageBlobs.js’ am definit obiectul ‘BLOB’, ce are legatura cu prezentarea grafica a proiectului si care reprezinta o colectie de puncte de proprietati asemanatoare, invecinate. In momentul in care utilizatorul incarca o imagine cu stele, algoritmul va procesa imaginea si va determina numarul lor. Stelele vor fi definite ca o multime de puncte, initial albe, ce se invecineaza la o distanta mai mica de 5 pixeli.

In fisierul ‘Interface.js’ am definit mai multe functii de incarcare a fisierelor prin intermediul ‘drag&drop’, fiecare corespunzand celor 3 optiuni ale programului.

Inainte de a pune cap la cap piesele pentru a prezenta fiecare algoritm, voi vorbi putin despre XOR, o functie ce nu poate fi separata printr-o linie dreapta. XOR este operatia logica ce retuneaza adevarat daca doar un element este adevarat.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| p | q | P AND q | P OR q | P XOR q |
| 1  1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 |

In momentul creeri primului Perceptron (Neuron), care prin natura sa poate aproxima functii separabile printr-o linie dreapta, s-a pus problema daca poate aproxima functii separabile prin doua linii dreapte. Deoarece un singur Perceptron nu era suficient, a aparut ideea de Retea de Perceptroni, adica Retea Neuronala, ce este capabila sa aproximeze o varietate mai mare de functii.

In timp ce operatiile logice ‘AND’ si ‘OR’ sunt separabile printr-o linie, ‘XOR’ este separabila prin 2 (Vedeti tabelul de mai sus). Fiind prima ecuatie aproximata cu ajutorul unei retele neuronale, aceasta a ramas problema clasica cu ajutorul careia se testeaza orice retea neuronala complexa. Voi folosi aceasta functie, XOR, ca exemplu pentru a prezenta functionalitatea si logica din spatele algoritmului.

Acest proiect este construit pe baza unei librarii numite ‘p5.js’. Aceasta are scopul de a face accesibil procesul de programare pentru artisti, designeri, profesori si incepatori. Astfel, libraria introduce functii pentru reprezentare grafica a formelor geometrice in plan si in spatiu. Importante sunt 2 functii, si anume ‘setup’, ce ruleaza o singura data, la deschiderea scriptului, si ‘draw’, ce ruleaza in fiecare cadru si care permite realizarea animatiilor. Alte 2 functii folosite sunt ‘keyPressed’ si ‘mousePressed’, ce sunt apelate in momentul apasarii unei taste si, respectiv, in momentul apasarii mouse-ului pe ‘canvas’. ‘Canvas’ este plansa, spatiul in care se deseneaza si se animeaza.

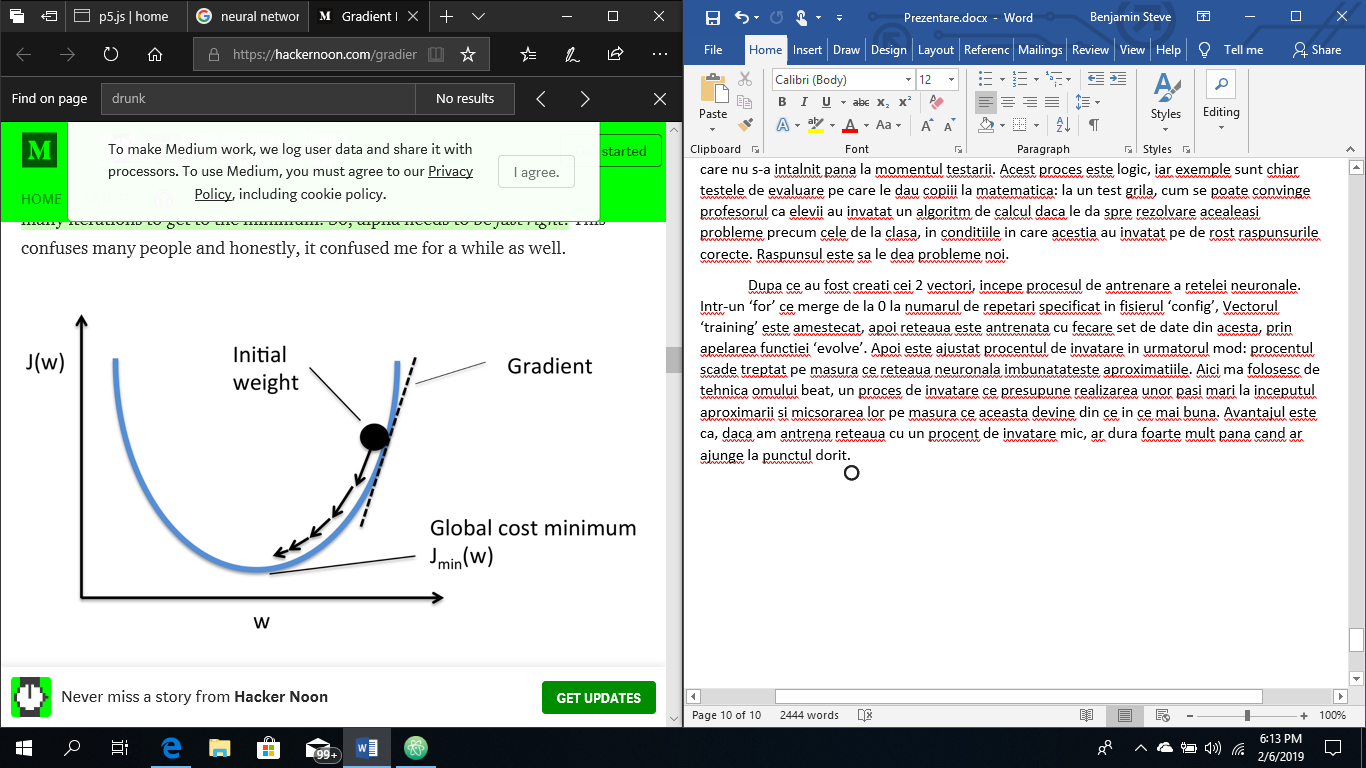
In momentul deschiderii fisierului in care se ruleaza proiectul, in acest caz ‘index.html’, utilizatorul va observa 3 optiuni: ‘Determina o ecuatie’, ‘Determina linia orizontului’, ‘Utilizeaza o ecuatie’. Acestea sunt afisate de catre functia ‘ChooseWhatToDo’ din ‘Interface.js’, apelata la inceputul programului, in functia ‘setup’. De aici, fiecarei optiuni ii corespund mai multe functii, ce sunt apelate in cascada.

Pentru prima optiune, si anume ‘Determina o ecuatie’, mai intai se creaza pe ecran un spatiu in care se vor introduce dimensiunile Retelei neuronale, adica numarul de noduri de intrare, ascunsi si de iesire. Apoi este apelata functia ‘UploadData’, ce creaza pe ecran o zona pentru ‘drag&drop’, prin care vor fi incarcate 3 fisiere: ‘variabile.txt’, ‘config.txt’, ‘rezultate.txt’. Continutul fiecarui fisier este salvat in cate o variabila. Deoarece procesul de incarcare nu se realizeaza instant, pot aparea erori daca am procesa imediat fisierele. Din acest motiv, am decis sa las o scurta pauza programului pentru a incarca fisierele. Dupa acest pas, este apelata functia ‘get\_data’, ce prelucreaza si salveaza in variabile noi informatia. Initial este procesat fisierul ‘config.txt’, salvat in variabila ‘config’, apoi sunt procesate si celelalte 2.

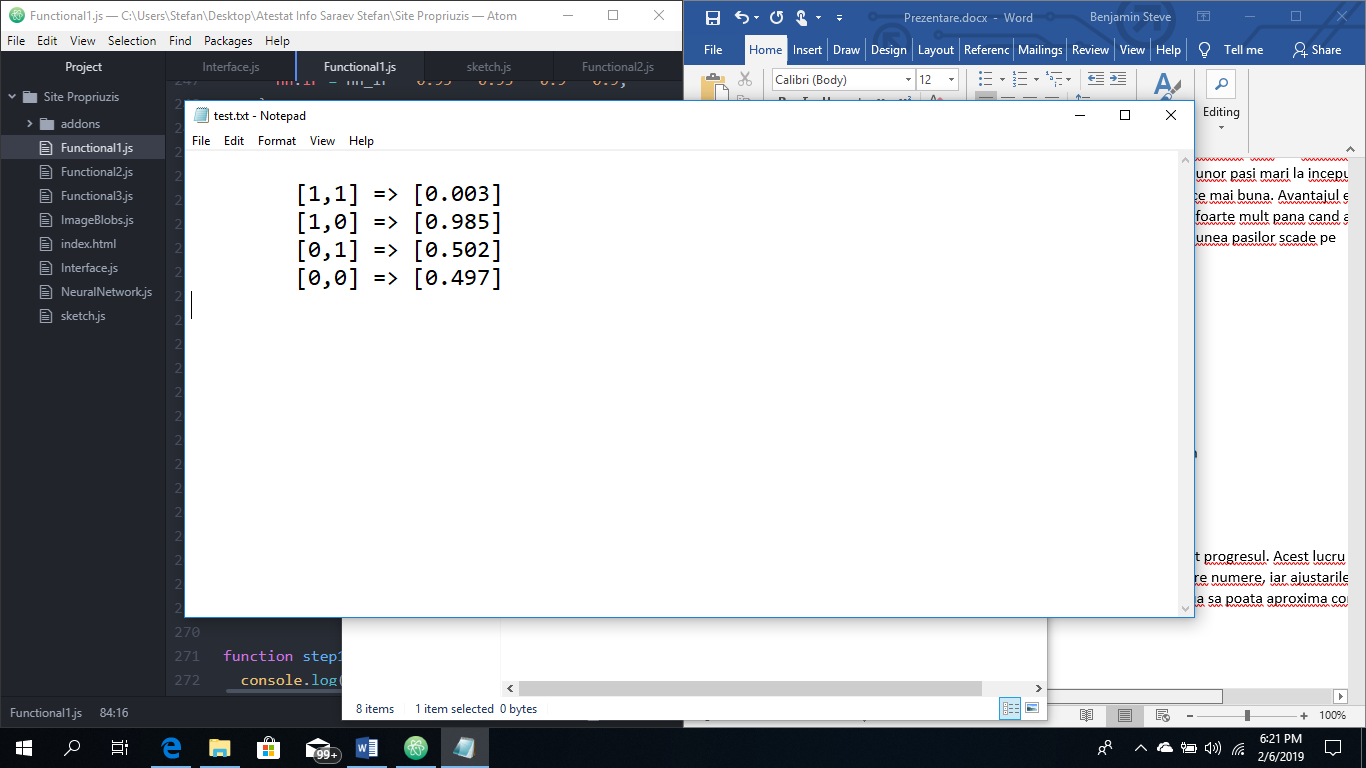
Procesarea este aproximativ aceeasi pentru toate fisierele si consta in urmatorii pasi: Descompunem variabila de tip String, in care este salvata informatia, mai intai dupa randuri, iar apoi dupa spatii (‘ ’). Fiecare cuvant rezultat ori este folosit ca marcator pentru o variabila, ori este convertit intr-un numar si salvat intr-o variabila. Pentru fisierul ‘config.txt’, marcatorii sunt cuvintele folosite in fisier pentru a reprezenta carei variabile ii va fi atribuita informatia ce urmeaza.

Dupa procesare, sunt create reteaua neuronala si doi vectori: ‘training’ si ‘testing’. Acesti vectori contin obiecte de tip ‘set’. Un set este multimea formata dintr-un set de date de intrare si datele de iesire corespunzatoare. Este necesar sa avem 2 vectori de seturi deoarece dorim sa antrenam reteaua cu mai multe seturi de date si sa o testam cu seturi noi de date, cu care nu s-a intalnit pana la momentul testarii. Acest proces este logic, iar exemple sunt chiar testele de evaluare pe care le dau copiii la matematica: la un test grila, cum se poate convinge profesorul ca elevii au invatat un algoritm de calcul daca le da spre rezolvare acealeasi probleme precum cele de la clasa, in conditiile in care acestia au invatat pe de rost raspunsurile corecte. Raspunsul este sa le dea probleme noi. Testarea va fi realizata apeland functia ‘predict’ pentru fiecare set de date de testare, iar daca raspunsul se afla in marginea de eroare acceptata, eficienta aproximarii creste.

Dupa ce au fost creati cei 2 vectori, incepe procesul de antrenare a retelei neuronale. Intr-un ‘for’ ce merge de la 0 la numarul de repetari specificat in fisierul ‘config’, Vectorul ‘training’ este amestecat, apoi reteaua este antrenata cu fecare set de date din acesta, prin apelarea functiei ‘evolve’. Apoi este ajustat procentul de invatare in urmatorul mod: procentul scade treptat pe masura ce reteaua neuronala imbunatateste aproximatiile. Aici ma folosesc de tehnica omului beat, un proces de invatare ce presupune realizarea unor pasi mari la inceputul aproximarii si micsorarea lor pe masura ce aceasta devine din ce in ce mai buna. Avantajul este ca, daca am antrena reteaua cu un procent de invatare mic, ar dura foarte mult pana cand ar ajunge la punctul dorit. Dupa cum se vede si in acest desen, dimensiunea pasilor scade pe masura ce reteaua neuronala se apropie de aproximatia cautata.



Daca programul nu reuseste aproximarea, va relua de la 0 tot progresul. Acest lucru se poate intampla deoarece retaua poate intelege partial legatura dintre numere, iar ajustarile vor deveni minime. In cazul aproximarii functiei XOR, se poate ca reteaua sa poata aproxima corect doar 2 seturi de date. In acest moment, se spune ca erorile aproximarii au atins un minim local, iar acesta poate fi depasit prin marirea procentului de invatare, astfel ca eroarea sa creasca, dar sa depaseasca acel minim. In cazul XOR, putem doar sa reluam procesul de la 0, intrucat avem doar 4 seturi de date de intrare.

 Daca programul reuseste aproximarea, se va genera un fisier cu extensia ‘.nn.txt’, ce va contine reteaua neuronala. Aceasta va fi folosita in optiunea a 3a.

Daca utilizatorul va alge optiunea a 3a, adica ‘Utilizeaza o ecuatie’, este apelata functia ‘UploadNN’, ce creeaza pe ecran o zona pentru ‘drag&drop’, cu ajutorul careia se vor incarca fisierul cu extensia ‘.nn.txt’ si un fisier ‘variabile.txt’.

Dupa ce a fost creata, la optiunea 1, reteaua neuronala, aceasta va fi folosita sa ofere niste rezultate aproximative pentru seturi de date noi. Practic, aplicabilitatea acestui proiect se rezuma la acest pas: utilizarea retelei generate in scopuri practice.

Dupa incarcarea celor 2 fisiere, acestea sunt procesate si este generat fisierul ‘rezultate.txt’, ce contine aproximari ale rezultatelor reale cautate, fisier ce se poate descarca.

Optiunea 2 reprezinta mai mult o demonstratie vizuala a aplicabilitatii acestui algoritm. Dupa ce este selectata aceasta optiune, se creaza pe ecran un spatiu in care se vor introduce dimensiunile Retelei neuronale, apoi va fi apelata functia ‘UploadImage’, ce va genera pe ecran o zona pentru ‘drag&drop’, cu ajutorul careia se va incarca un fisier cu extensia .jpg’. Aceasta imagine va contine, pe un fundal negru, mai multe puncte albe. Dupa incarcarea imaginii, aceasta va fi procesata si se vor extrage toate multimile de pixeli invecinati in obiecte ‘BLOB’. Prin apasarea pe stele, acestea se vor colora in culoarea afisata in stanga sus. Pentru schimbarea culorii, se va apasa pe tasta ‘R’ pentru rosu si pe tasta ‘V’ pentru verde. Dupa ce au fost colorate stelele, se va crea o retea neuronala ce va trebui sa separe cele 2 culori. Intrucat reteaua trebuie sa separe doar stelele de pe ecran, vectorul ‘testing’ va fi identic cu vectorul ‘training’. Procesul de antrenare este identic cu cel de la Opriunea 1. Dupa ce s-a terminat, sunt trasate una sau mai multe linii ce vor separa cele 2 tipuri de stele.

# 3. MANUAL DE UTILIZARE

Pasul 1: Dezarhivati fisierele

Pasul 2: Deschideti *Site Propriuzis/index.html*

Pasul 3: In meniu, alegeti una din optiuni. Mai jos voi folosi urmatoarele notatii:

*O1 = Determina o ecuatie*

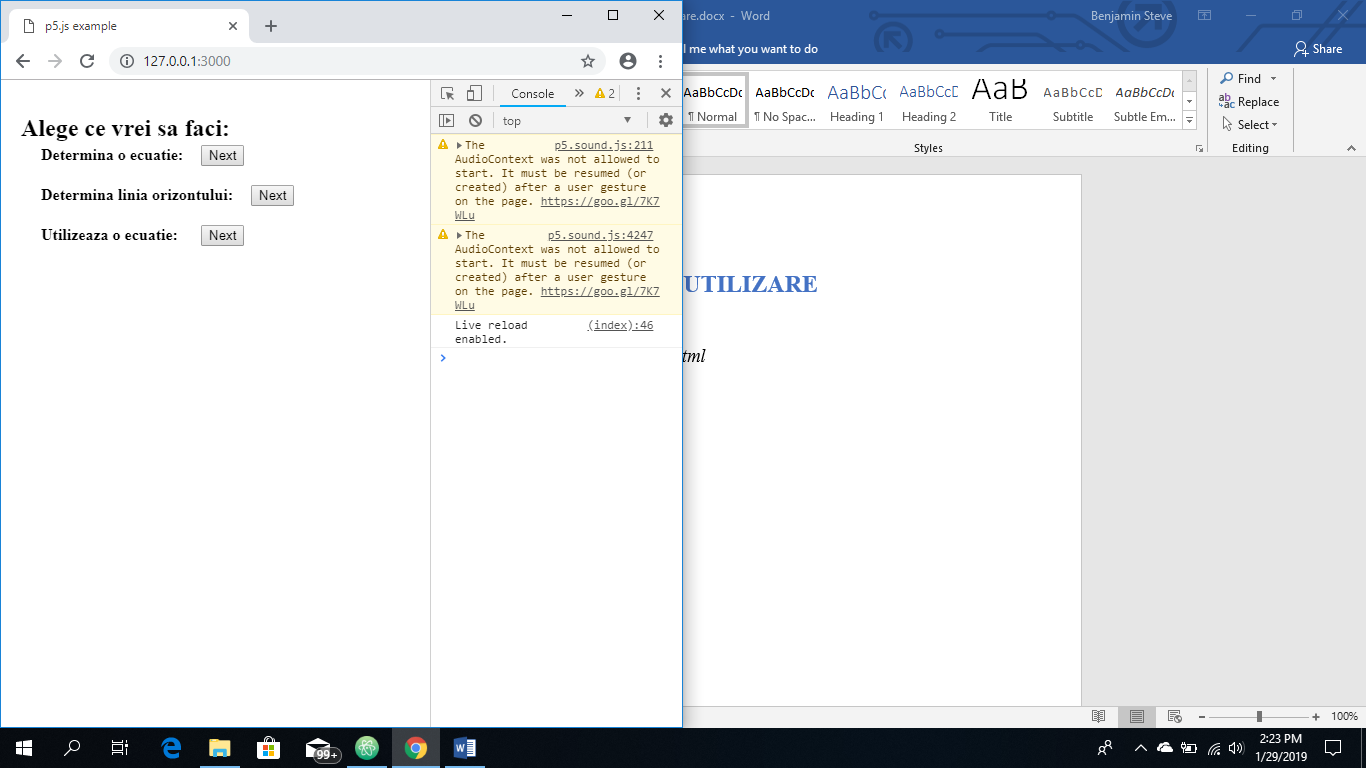
*O2 = Determina linia orizontului*

*O3 = Utilizeaza o ecuatie*

O1: Aici veti crea o aproximare a unei ecuatii sau a unei clasificari, urmand ca sa o descarcati si folositi in optiunea O3

O2: Aici veti observa un exemplu de clasificare a unor date dupa un criteriu: culoarea.

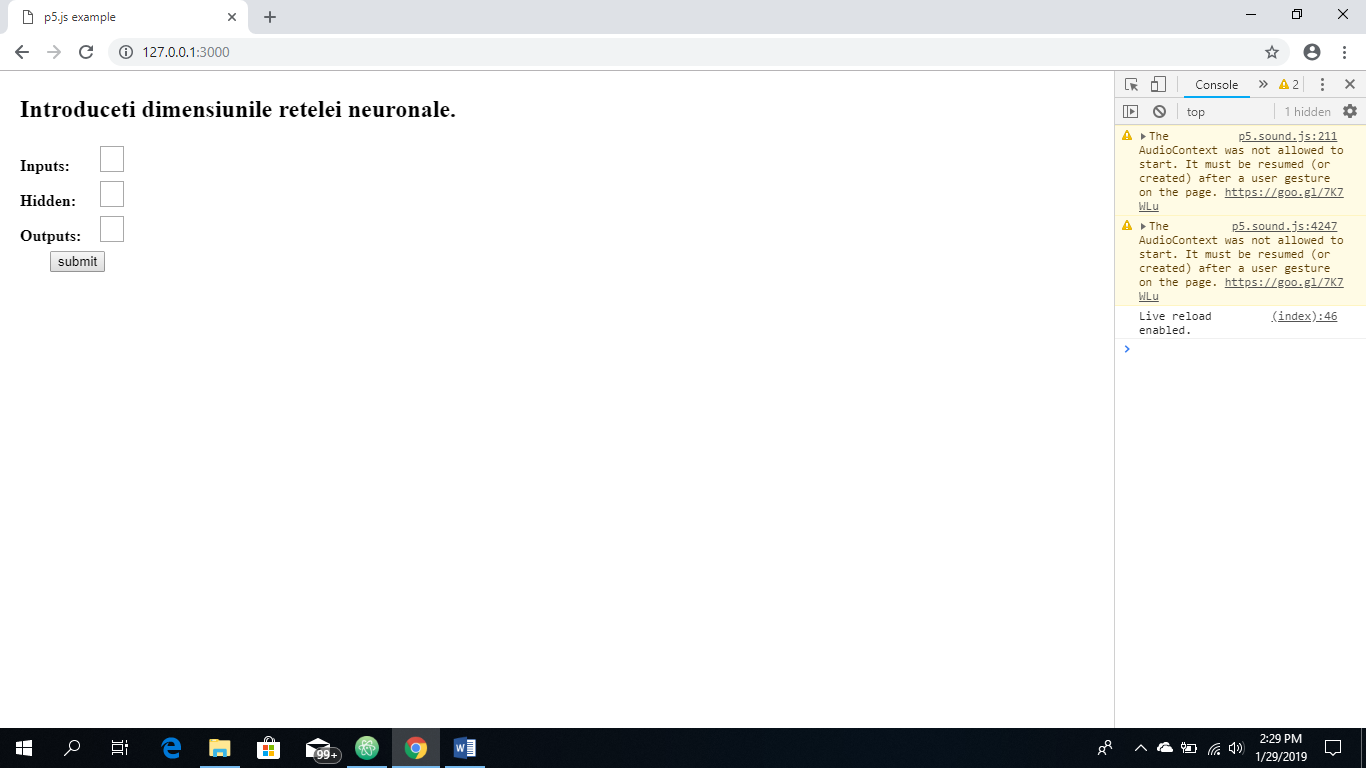
O3: Aici veti folosi ce ati obtinut de la O1 pentru a determina rezultatele ecuatiei aproximate pentru niste variabile/seturi de date noi.



## Optiunea 1

O1.Pasul 1: Apasati *Next*

O1.Pasul 2: Introduceti valorile pentru campurile *Inputs, Hidden, Outputs*, apoi apasati *submit*.

In campul **Inputs** se va trece numarul de variabile de intrare.

In campul **Hidden** se va trece un numar ce va reprezenta complexitatea aproximatiei.

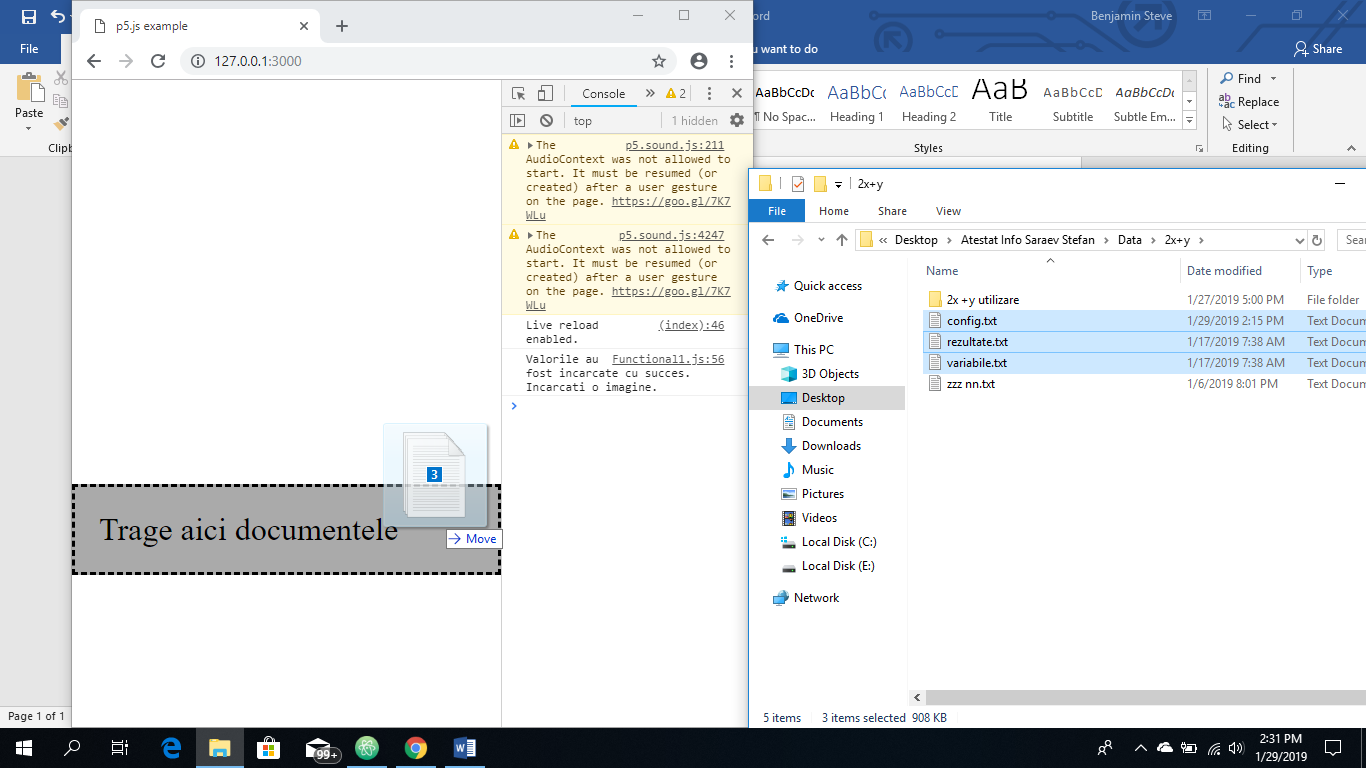
In campul **Outputs** se va trece numarul de variabile de iesire.

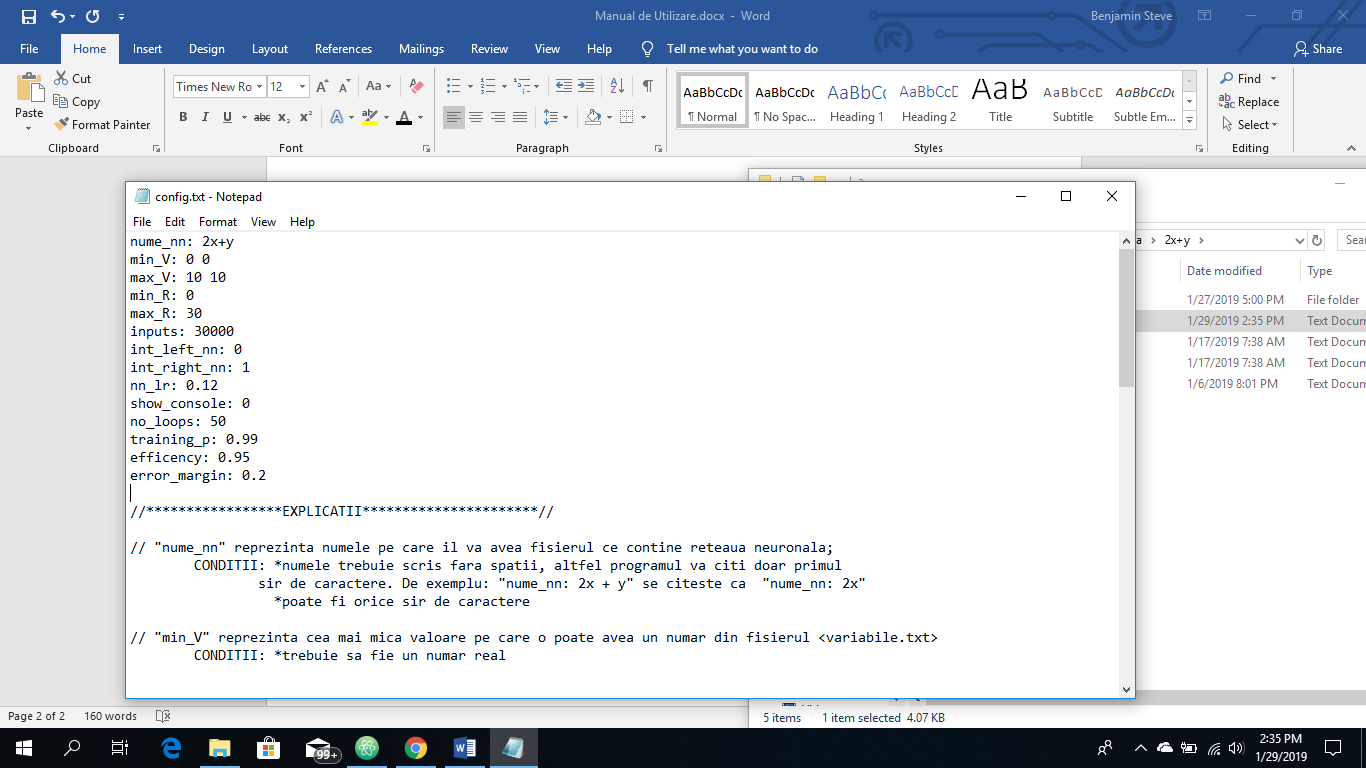
O1.Pasul 3: Incarcati prin *Drag & Drop* fisierele *variabile.txt, rezultate.txt, config.txt*.

In fisierul **variabile.txt** se vor afla seturi de date de intrare pentru care se stiu datele de iesire. Vezi Anexa1 pentru mai multe informatii.

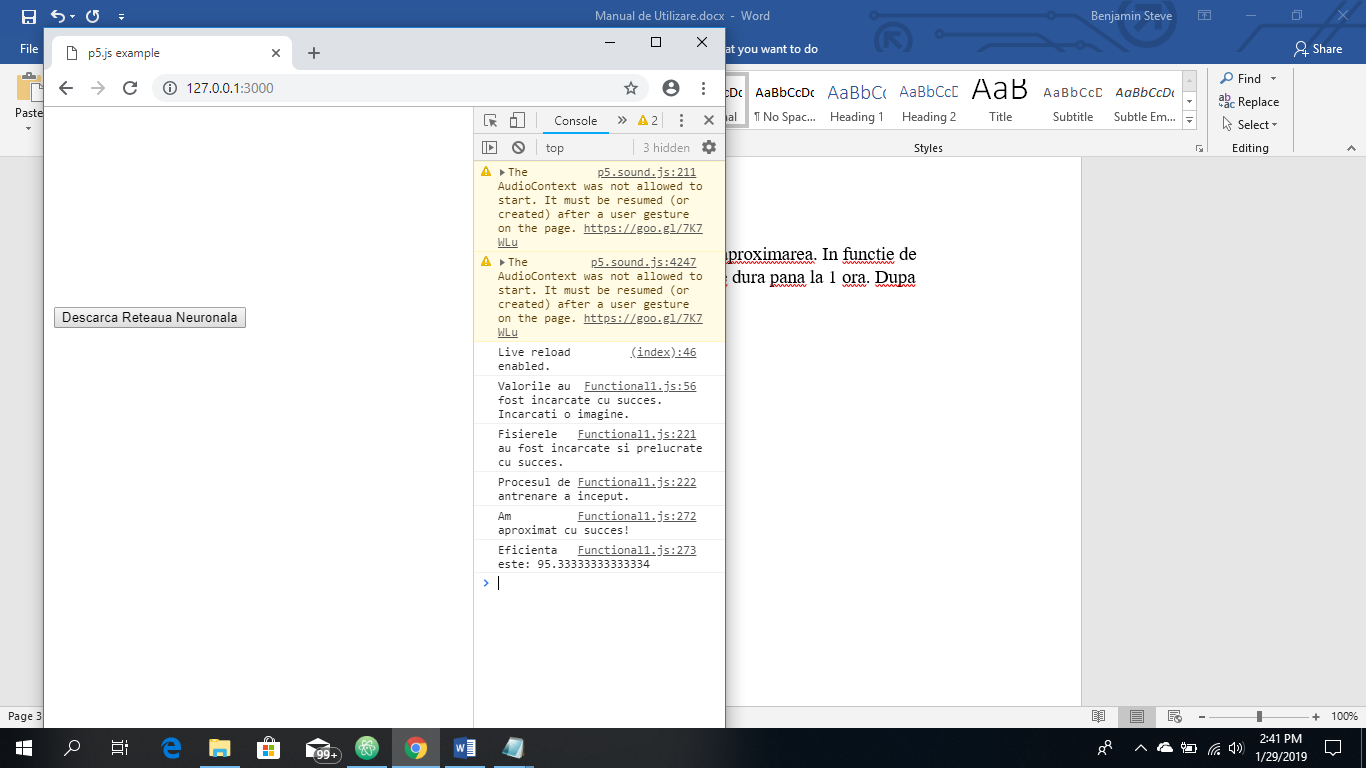
In fisierul **rezultate.txt** se vor afla seturile de date de iesire corespunzatoare celor din **variabile.txt**.

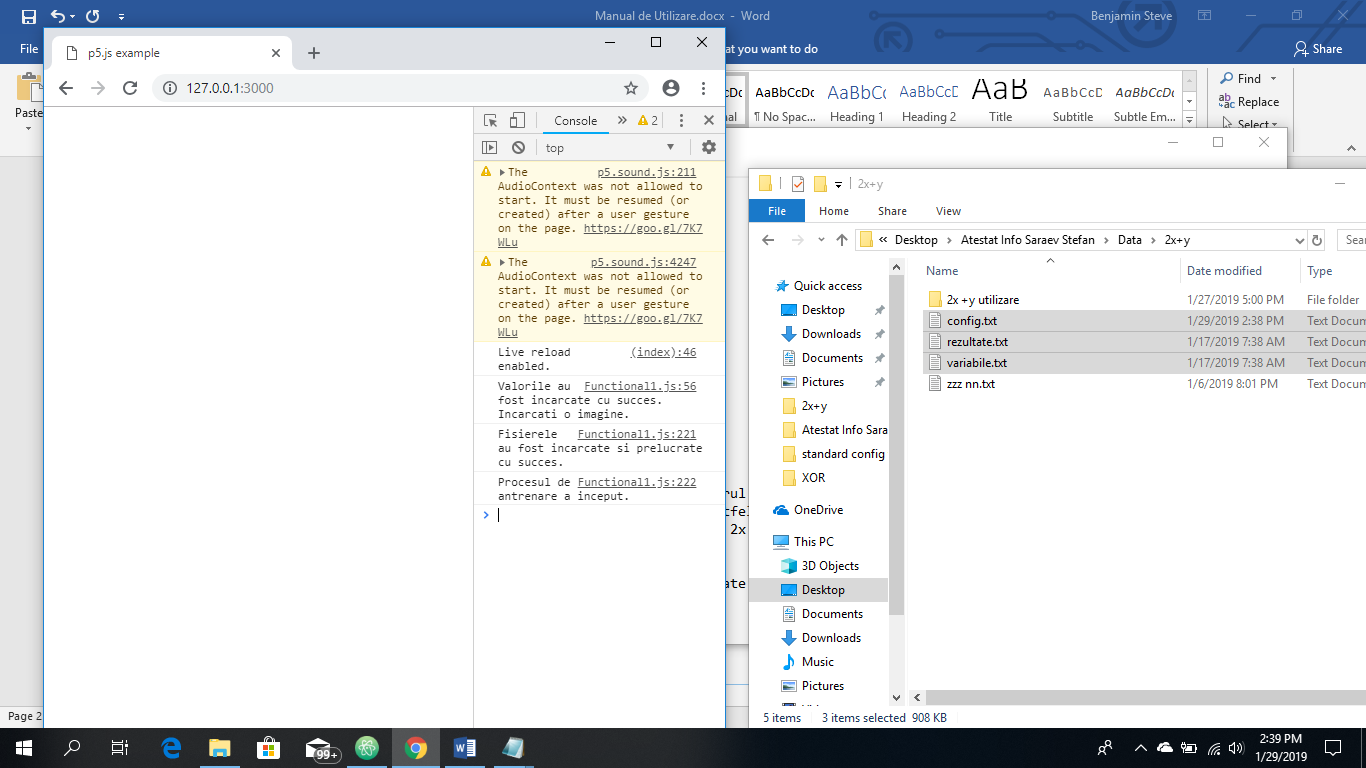
In fisierul **config.txt** se vor afla mai multe variabile introduse de la tastatura, corespunzatoare fiecarei aproximatii. Ordinea variabilelor nu conteaza. Vezi Anexa2 pentru mai multe informatii.





O1.Pasul 4: Asteaptati pana cand se finalizeaza aproximarea. In functie de complexitatea ecuatiei, aceasta aproximare poate dura pana la 1 ora. Dupa finalizare, apasati pe *Descarca Reteaua Neuronala*.





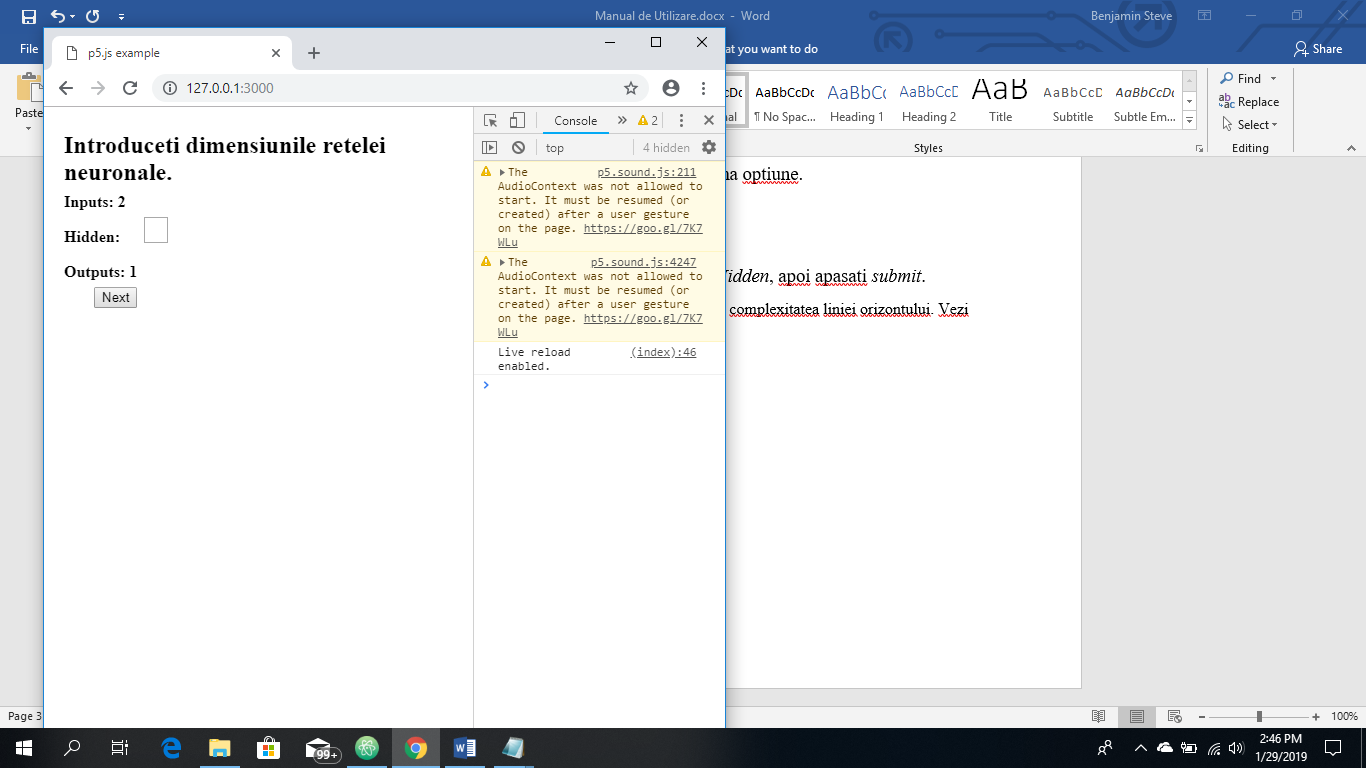
O1.Pasul 5: Creati un folder nou in care veti introduce fisierul descarcat.

FELICITARI! Ati invatat cum sa operati cu prima optiune.

## Optiunea 2

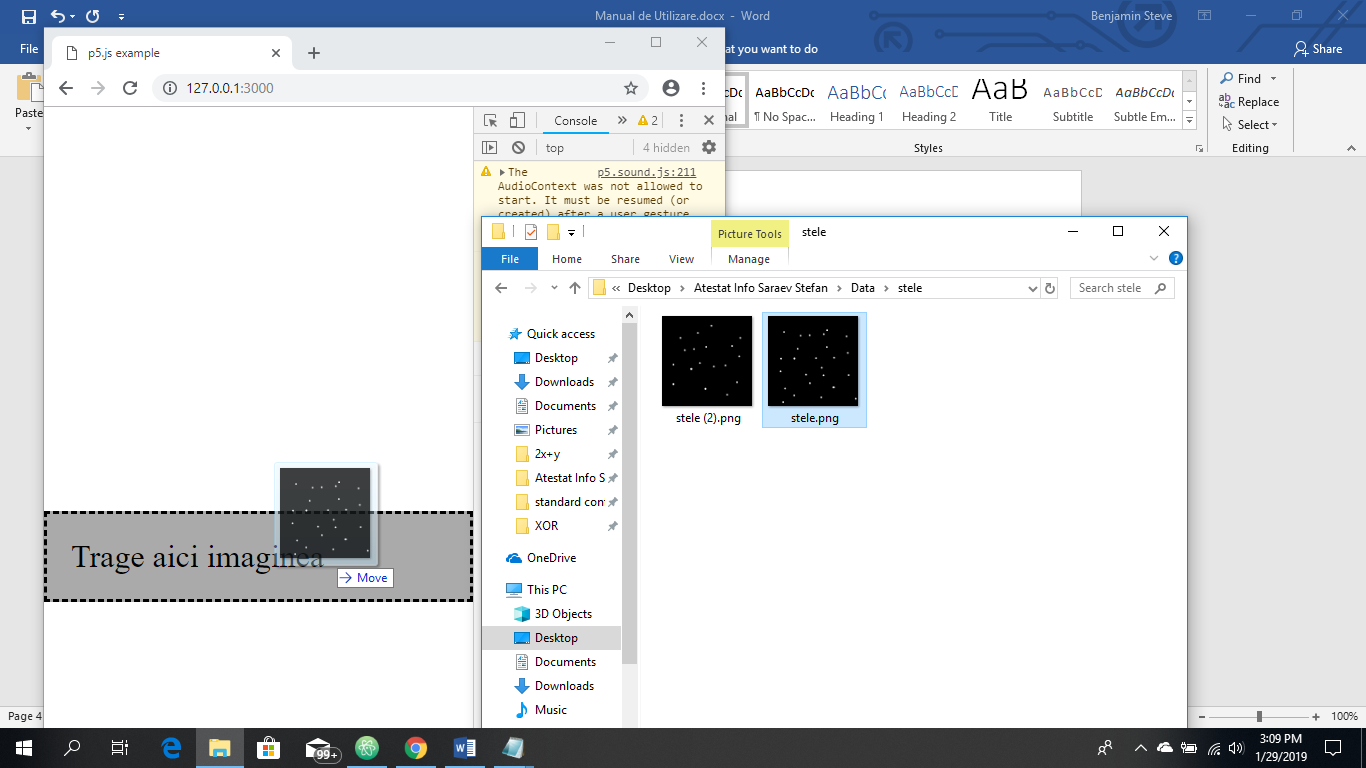
O2.Pasul 1: Apasati *Next*.

O2.Pasul 2: Introduceti valorile pentru campul *Hidden*, apoi apasati *submit*.

In campul **Hidden** se va trece un numar ce va reprezenta complexitatea liniei orizontului.

O2.Pasul 3: Incarcati prin *Drag & Drop* o imagine *.png.*

Exista 2 imagini un acest proiect. Acestea se afla in directorul *Data/stele*.

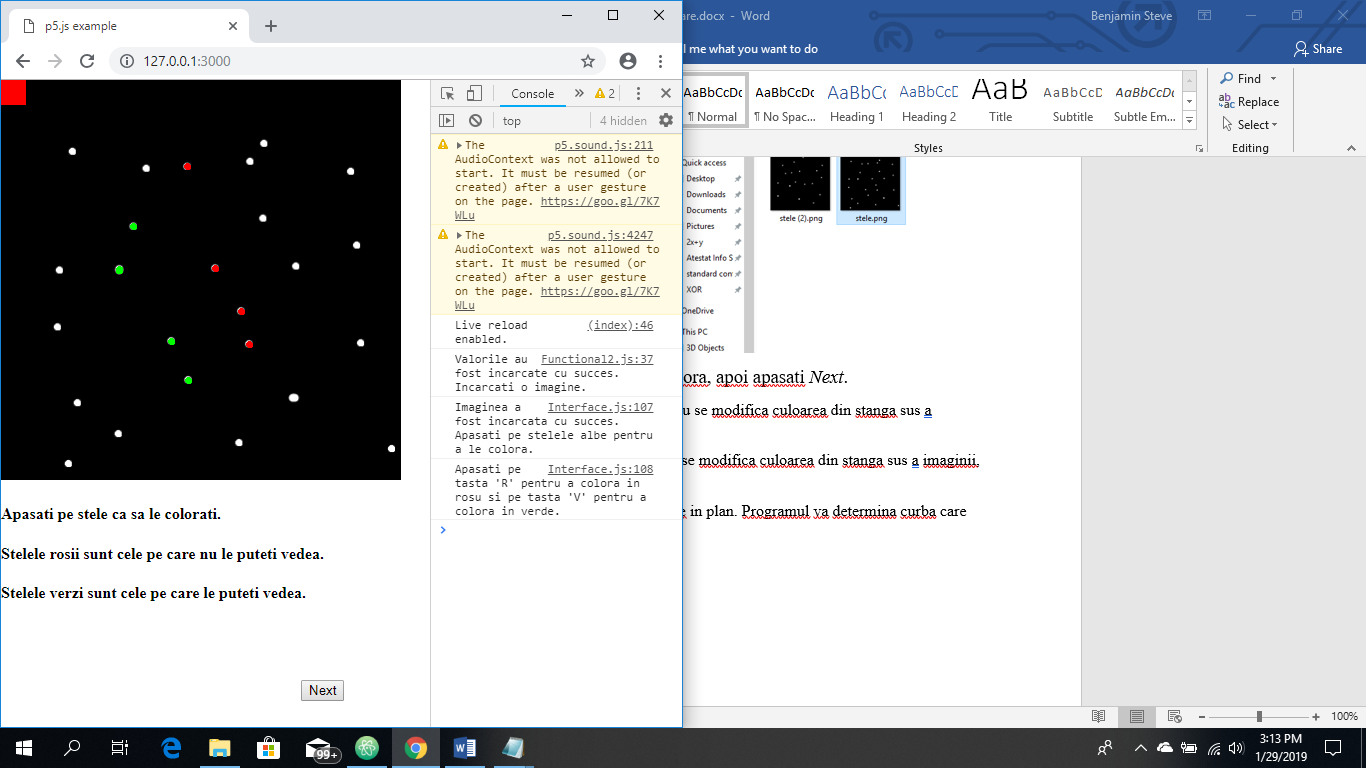


O2.Pasul 4: Apasati pe stele pentru a le colora, apoi apasati *Next*.

Pentru a colora cu verde, apasati tasta ‘V’. Daca nu se modifica culoarea din stanga sus a imaginii, apasati ‘CapsLock’ si apoi tasta ‘V’.

Pentru a colora cu rosu apasati tasta ‘R’. Daca nu se modifica culoarea din stanga sus a imaginii, apasati ‘CapsLock’ si apoi tasta ‘R’.

Aceasta colorare este o clasificare a unor elemente in plan. Programul va determina curba care separa cele doua tipuri de stele, rosi si verzi.



O2.Pasul 5: Programul va afisa o curba galbena ce va delimita cele doua tipuri de stele.

Daca apare in consola “Reiau procesul de la 0.” de mai mult de trei ori inseamna ca complexitatea curbei este mai mare decat anticipata. In acest caz, dati Refresh la pagina si incepeti din nou de la Pasul 1, urmand ca la Pasul 2 sa introduceti o valoare mai mare in campul **Hidden**.

FELICITARI! Ati invatat cum sa operati cu a doua optiune.

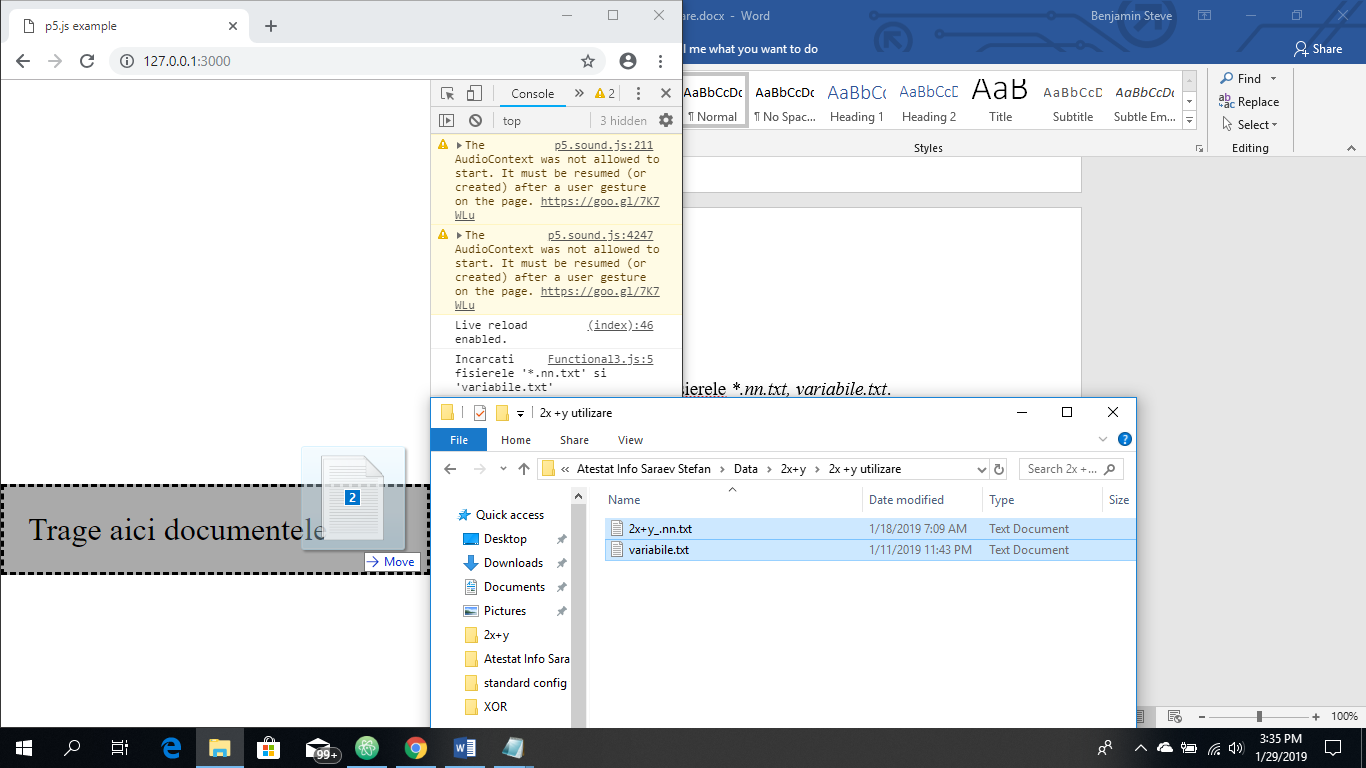
## Optiunea 3

O3.Pasul 1: Apasati *Next*

O3.Pasul 2: Incarcati prin *Drag & Drop* fisierele *\*.nn.txt, variabile.txt*.

Fisierul **\*.nn.txt** este cel descarcat de la **O1.Pasul 5**

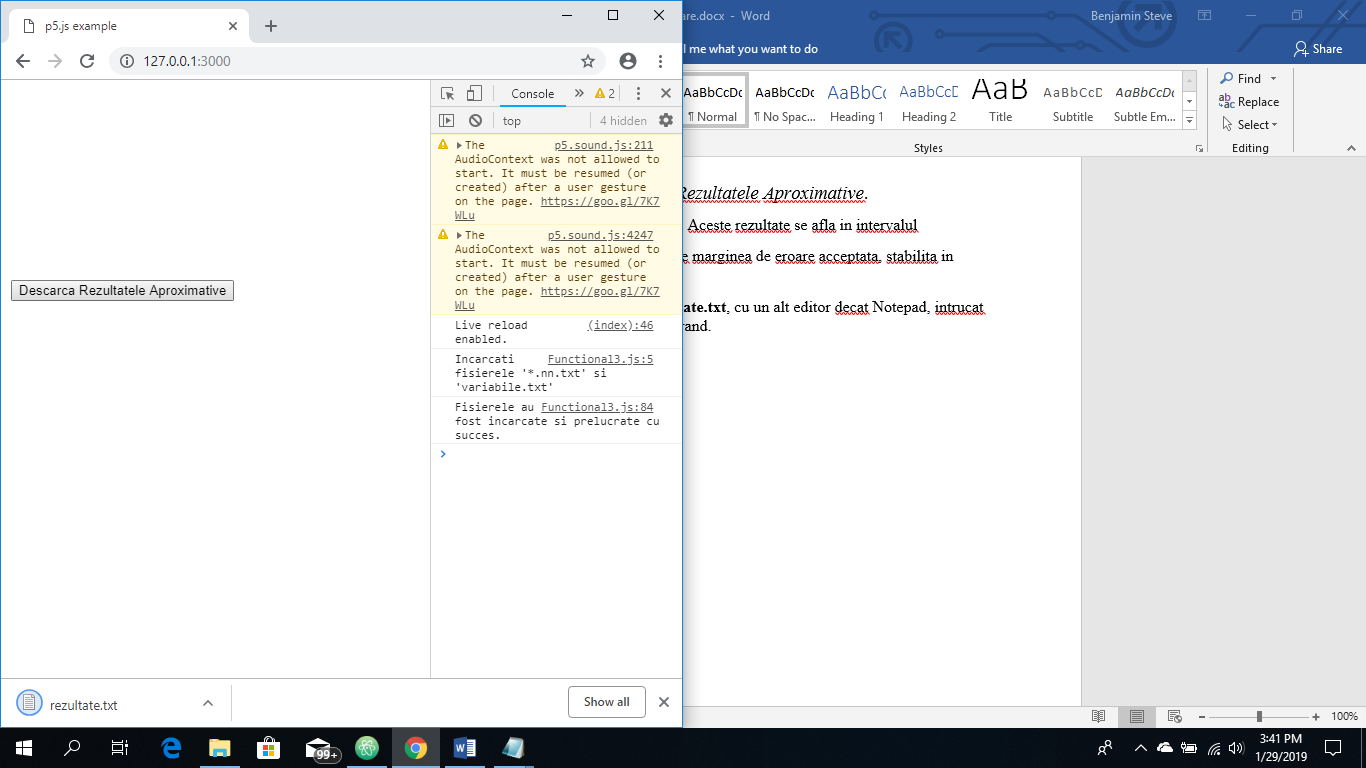
In fisierul **variabile.txt** se vor afla seturi de date de intrare pentru care dorim sa stim datele de iesire. Acest fisier se adauga in folderul unde se afla si **\*.nn.txt**.

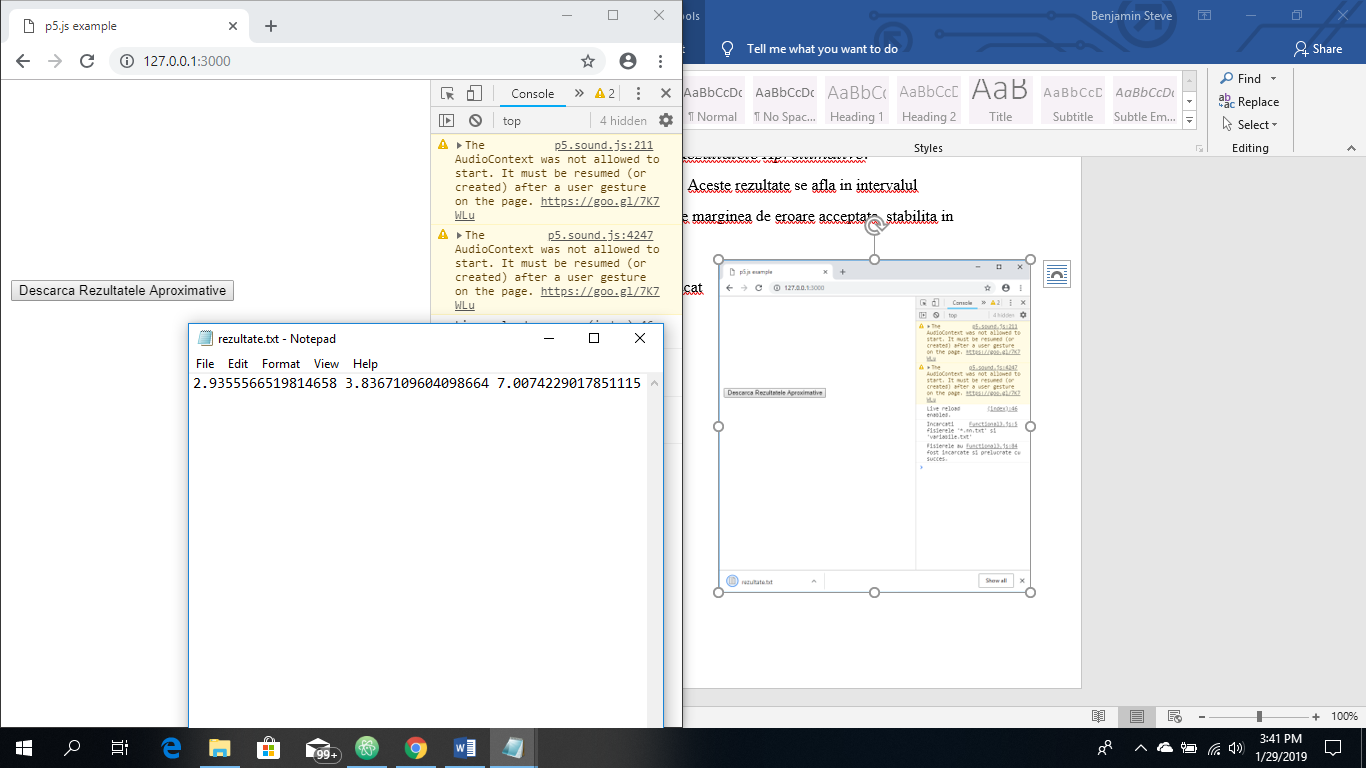
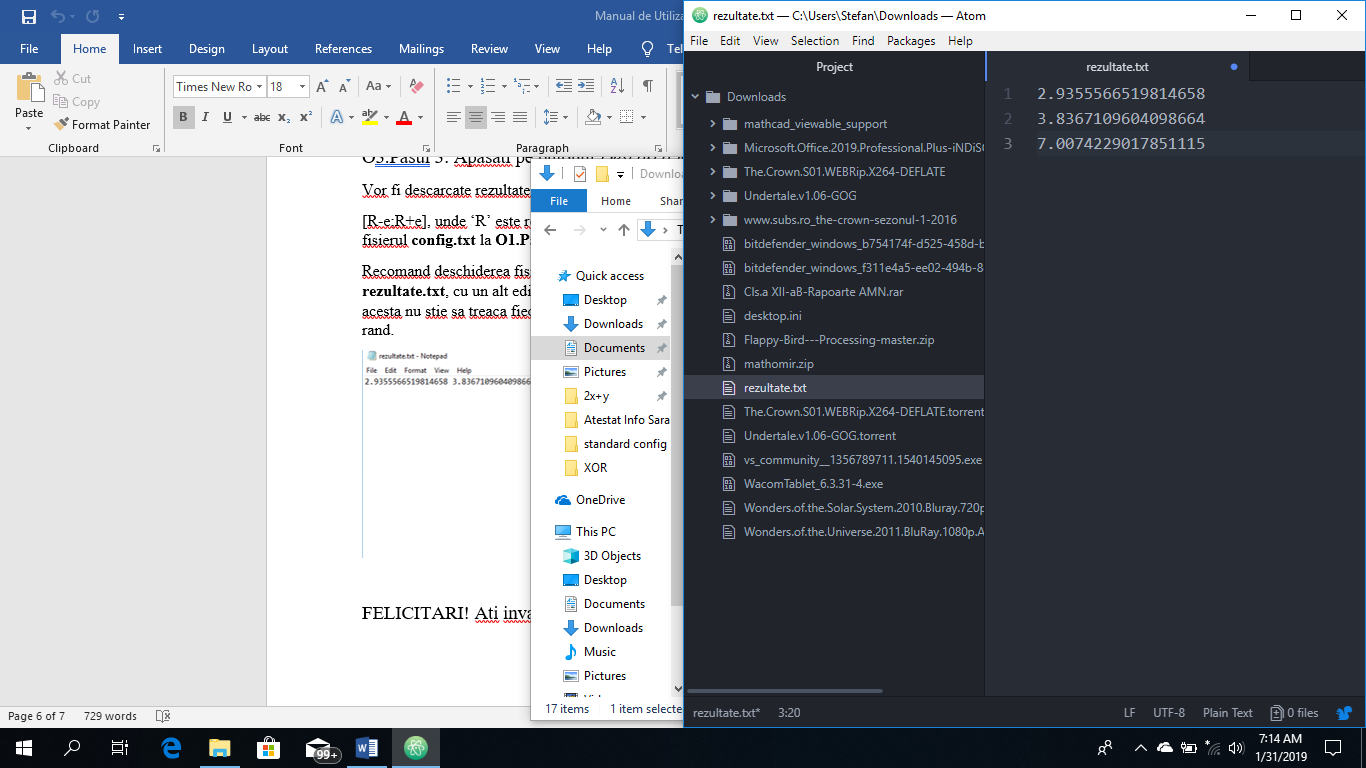


O3.Pasul 3: Apasati pe butonul *Descarca Rezultatele Aproximative*.

Vor fi descarcate rezultatele aproximative cautate. Aceste rezultate se afla in intervalul

[R-e;R+e], unde ‘R’ este rezultatul real, iar ‘e’ este marginea de eroare acceptata, stabilita in fisierul **config.txt** la **O1.Pasul 3**.

Recomand deschiderea fisierului descarcat, **rezultate.txt**, cu un alt editor decat Notepad, intrucat acesta nu stie sa treaca fiecare rezultat pe cate un rand.



FELICITARI! Ati invatat cum sa operati cu a treia optiune.

# 4. BIBLIOGRAFIE

* Manualul de Informatica, clasa a XII-a, editura Corint, Autori: Mioara Gheorghe, Monica Tataram, Corina Achinca, Ioana Pestritu
* <https://p5js.org/reference/>
* https://thecodingtrain.com/
* <https://hackernoon.com/gradient-descent-aynk-7cbe95a778da>

# 5. ANEXE

Anexa1: In fisierele **variabile.txt** si **rezultate.txt** se vor scrie numerele dintr-un sed de date unul dupa altul, cu spatiu intre ele. Seturile de date se separa prin linie noua (tasta ‘ENTER’).

De exemplu:

3 variabile de intrare, 2 de iesire, 4 seturi de date

variabile.txt: rezultate.txt:

1 3 5 1 4

2 5 7 2 6

3 1 2 3 1

7 5 6 7 5

Anexa2: In fisierul **config.txt** se vor introduce variabilele pe baza carora va fi antrenata reteaua neuronala spre a aproxima o ecuatie. In acest fisier, ordinea variabilelor nu contreaza, cat timp este respectata urmatoarea regula: Orice instructiune din fisierul config va fi sub forma:

[numele variabilei]: [valoare/valori]

De exemplu:

nume\_nn: 2x+y

max\_V: 10 10

In fisierul **config.txt** este explicat fiecare camp individual.