Miina Härma Gümnaasium

Kristjan Koitla

**ISEÕPPIV TEHISINTELLEKT**

Praktiline töö

Juhendaja: Allar Aav

Matemaatika õpetaja

Tartu 2018

## **SISUKORD**

[SISUKORD 2](#_Toc515875151)

[SISSEJUHATUS 3](#_Toc515875152)

[ÜLDISELT 5](#_Toc515875153)

[1. PROGRAMMI TÖÖPÕHIMÕTE 6](#_Toc515875154)

[2. PROGRAMMI VALMIMISPROTSESS 9](#_Toc515875155)

[2.1 Teema mõistmine 9](#_Toc515875156)

[2.2 Arendus 10](#_Toc515875157)

[2.3 Keele valimine ja õppimine 11](#_Toc515875158)

[2.4 Programmeerimine 12](#_Toc515875159)

[3. TULEMUSED 13](#_Toc515875160)

[3.1 Programmi testimine 13](#_Toc515875161)

[3.2 Praktilise töö tulemused 15](#_Toc515875162)

[KOKKUVÕTE 16](#_Toc515875163)

[KASUTATUD KIRJANDUS 17](#_Toc515875164)

[SELF-LEARNING ARTIFICIAL INTELLIGENCE 18](#_Toc515875165)

# 

## **SISSEJUHATUS**

Tarkvaratehnoloogiast huvituva gümnasistina oli lihtne panna tähele masinõppe teemalisi projekte internetis, mis tekitasid huvi. Huvist teema vastu sai teada üks teine iseõppiv süsteem – Google, mis tänu sellele teemakohaseid õppevideosi reklaamima hakkas. Idee, et üks iseõppinud süsteem soovitab kellelegi masinõppe õppevideosi oli nii surreaalne ja samas haarav, et sai teemaga teha lähemalt tutvust. Sageli on artiklid ja videod õppe kohta tehtud selliselt, et uuele huvitundjale võib sarnase projekti ise tegemine tunduda võimatu, kuid kontsepti uurides selgub, et see ei vasta tõele. Olles vaimustuses ideest, et ka töö raames võiks midagi sellist luua, sai vastu võetud otsust, et just masinõppest praktiline töö teha.

Tuginedes viimase aja tehnoloogiaalastele ajakirjadele on märgata selget masinõppeuudiste kasvavat populaarsust, millest võib järeldada, et teema on praegu ülemaailmselt kasvav huvi. Teema populaarsuse tõusule annab tunnistust veel see, et teemat on Google otsinguga otsitud peaaegu 10 korda rohkem kui 5 aastat tagasi.1

Sellise iseõppiva süsteemi jaoks oleks kõige lihtsam kasutada juba ette tehtud raamistike nagu Tensorflow või Microsoft CNTK, mis on tasuta ja mille kasutamine ei nõua isegi täielikku teema valdamist. Eelolevate raamistike kasutamine teeks küll protsessi palju lihtsamaks, kuid töö eesmärk ei ole teha ümber kellegi teise koostatud süsteemi, vaid mõista, kuidas iseõppivad süsteemid töötavad ja iseseisvalt üks selline algusest lõpuni teha.

Nii langes otsus õppida masinõppe olemust ja siis kirjutada algoritm algusest, sest selline töökäik annab kõige põhjalikuma ülevaate teemast ja sellest, kuidas õpe toimib, ja ilma välise abita algoritmi tegemine tundus kõige ausam. Algoritm pidi ka tõestama enda reaalset õppimisvõimet.

Arvestades töö mahukust ja programmi keerukust, sai otsustatud, et programm tuleb võimalikult efektiivne ning kood loetav ja samas kompaktne. Praktilisel tööl kujunes lõpuks välja viis etappi: esiteks tuli leida internetist õppematerjal ja selgeks teha masinõppe printsiibid ja loogika, teiseks tuli teadmised panna kasutusse ja arendada algoritm, kolmandaks oli vaja valida sobilik keel, milles programm kirjutada ja peale seda tuli programm kirjutada. Kõige viimane etapp oli algoritmi proovilepanek ja tõestamine, et programm on õppesuutlik.

## **ÜLDISELT**

Tom M. Mitchell on defineerinud masinõpet lausega “*Arvutiprogramm õpib kogemusest E ülesannete klassi T ja tulemuste mõõdu P suhtes, kui selle tulemuslikkus klassi T ülesannetes, nii nagu mõõdetud P kaudu, paraneb kogemusega E.*” 2

Lihtsamalt ümber sõnastades tähendab see seda, et programmi põhiline eesmärk on teha üldistusi varasemate kogemuste põhjal ja parendada nende põhjal järgnevaid tulemusi.

Tehisintellekti iseõppimine sai alguse 1950. aastatel kabet mängiva programmiga, mille autoriks oli Robert Kräm. 1960. Aastatel loodi esimene tehislik “närvivõrk” pertseptron, mis teisendab sisendi väljundiks. 1970. Paarkümmend aastat hiljem hakkasid populaarsust koguma tehislikud närvivõrgud ja arendati tekstiõppimist, andmekaevet ja induktiivsete loogikaülesannete lahendamist. Viimase aastakümnega on arendatud graafilisi mudeleid, masinõppimist robootikas ja masinnägemises. 3

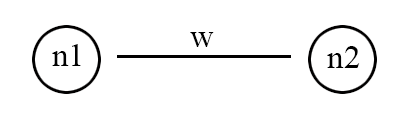
Meetodeid, kuidas tarkvara ise õppima panna on palju, kuid nendest kõige suurema potentsiaaliga on tehislikud närvivõrgud ja süvaõpe. Mõlemad meetodid on inspireeritud evolutsiooni käigus arenenud ehtsate närvisüsteemide poolt. Mõtlemine põhineb tehislikel neuronitel, mis on üksteisega ühendatud.

Tänapäeval on masinõpe kasutuses laialdaselt paljudes erinevates valdkondades nt. liiklusfoorides, reklaamide isikustamises, tervisehoius jne. Kõige silmapaistvamateks näideteks võib pidada Google otsingut, isesõitvaid autosi ja erinevaid tõlkeprogramme.

## 

## **1. PROGRAMMI TÖÖPÕHIMÕTE**

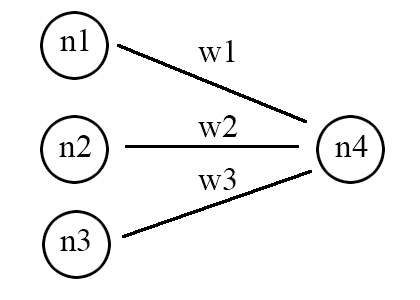
Evolutsioonil on lihtne printsiip, kus osavad jäävad ellu ning paljunevad ja vähem osavad mitte. Sarnaselt töötab töö raames tehtud süsteem, mis genereerib suure populatsiooni varieeruvate omadustega üksusi, kes pannakse ühekaupa proovile. Kõige vähem edukad sooritajad kustutatakse ja asendatakse kõige edukamate sooritajate duplikaatidega. Peale nõrkade tugevamate vastu välja vahetamist, muudetake juhuse saatel igat üksust nii, et mitte ükski üksus ei oleks samasugune teisega, ehk siis muteeritakse.

Järgmine loogiline küsimus oleks: mida muteeritakse üksuses või kuidas üksus teeb sisenditest väljundid ehk kuidas mõtleb üksus? Iga üksus koosneb tehisneuronitest, mis on üksteisega ühendatud. Igal tehisneuronil ja sidemel on oma väärtus, mille põhjal arvutatakse järgmine neuron. Iga järgmine neuron saab oma väärtuse eelmiselt neuronilt, kuni jõutakse viimaseni - üksuse väljundini. Kõige esimeste neuronite väärtused saadakse vastavalt ülesandele või keskonnale, mida üksused õppima peavad. Näiteks võivad esimesed väärtused olla kaks numbrit, mida üksus peab liitma.

Joonis 1. Kaks neuronit ning nende vaheline suhe.

Joonisel on visualiseeritud kahte neuronit, n1 ja n2, ja nende vahelist suhet, w. Siin saab n2 enda väärtuseks n1 ja w korrutise, mis käib veel läbi spetsiaalse tasakaalustamisfunktsiooni, et järgmine neuron ei saaks ebamääraselt suuri või väikseid väärtusi. Praeguseks näiteks võib tuua funktsiooni ArcTan, mis ei lase numbril muutuda liiga suureks ega ka liiga väikseks, et järgmine neuron seda kasutada saaks.

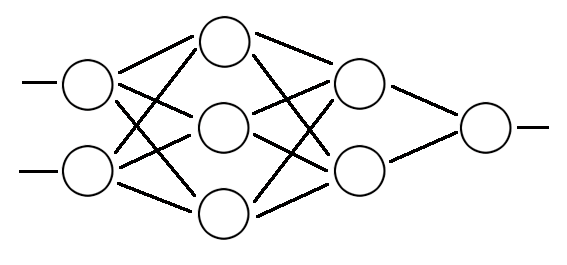
Eelneval joonisels saadakse teise neuroni väärtus kasutades valemit “*n2 = ƒ(n1) \* w*”, kus *ƒ(x)* on funktsioon ArcTan.

Reaalsuses on väga haruldane leida neuronit, mis on ühenduses ainult ühe teise neuroniga, kuna selline suhe oleks ebaefektiivne. Kui ühendunud neuroneid on aga rohkem, leitakse kõikide nende suhete korrutiste summade keskmine või summa, olenevalt neuronvõrest.

Joonis 2. Ühe neuroni suhe kolme teise neuroniga.

Eelneva joonise viimase neuroni väärtus saadakse valemiga “*n4 = ƒ(n1)\*w1 + ƒ(n2)\*w2 + ƒ(n3)\*w3*” , kus *ƒ(x)* on funktsioon ArcTan.

Enamasti on sellised neuronvõred suuremad, kuigi nende suurused võivad olla väga varieeruvad.



Joonis 3. Täielik neuronvõre 8 neuroniga.

Joonisel on visualiseeritud täielikku neuronvõre, kus esimesed kaks otsteta sidet tähistavad üksuse sisendit ja viimane otsteta side tähistab üksuse väljundit. Selline neuronvõre suudab näiteks väga täpselt võrrelda kahte numbrit või teha lihtsamaid tehteid. Iga selline võre saab nt. kas liita, lahutada, korrutada või jagada kahte arvu.

Iga neuroni kiht arvutatakse ühekaupa ja igas kihis arvutatakse iga neuron ühekaupa, kuni jõutakse väljundini. Üksuseid saab muteerida muutes neuronite vaheliste sidemete ehk suhete väärtusi. Kokkuvõttes “mõtleb” iga üksus läbi matemaatiliste tehete ja areneb läbi muteerumise.

# 

## **2. PROGRAMMI VALMIMISPROTSESS**

### **2.1 Teema mõistmine**

Esimene samm projekti juures oli üks tähtsamaid, nimelt teema mõistmine. Õnneks on internetis palju õppemeediume, mille seast valida. Oli teada algusest, üks parimaid meetodeid internetis õppimiseks on kursused, sest viimastega on autoril olnud head kogemused.

Veebipõhiseid kursusi masinõppe jaoks oli üllatavalt palju. Kursust otsides oli soov leida midagi, kus on nii õppevideod, kui ka kontrollküsimused. Õiget kursust otsides, oli näha tuttavat nime, milleks oli Google. Ettevõte, mis on teinud enda nime just tänu masinõppe saadustele, sai tähelepanu ja lõpuks ka poolehoiu.

Kursus asus veebilehel Udacity ja oli tasuta. Sellega õppimine oli tehtud meeldivaks, täpsemalt oli seal küllalt videosi, loovaid ülesandeid ja kordavaid vahekontrolle. Terve kursuse projekteeritud aeg oli kolm kuud, kuid mingi masinõppe kogemus oli juba varem olemas ja kursuselt sai valida välja kõige sobilikumad alateemad, seega teema piisavaks valdamiseks läks paar nädalat.

Masinõppe õppimisele aitas kaasa ka see, et oli olemas ka programmeerimise tunnis sellega tegelemiseks luba, kuna tundides käsitletud teemad olid juba varem omandatud. Peale selle oli Udacity ja selle kursused ka nutitelefonil saadavad, seega ei pidanud olema arvutis, et kursusest osa võtta. Vahepeal sai ka kooli vahetundidel teemat mugavalt telefonist uurida. Isegi peale kursuse läbimist, sai projekti tegemise käigus sageli midagi uut teema kohta teada.

### **2.2 Arendus**

Kursuselt kõik vajaliku kaasa saanud, tuli arendada algoritm ja mõelda välja, kuidas seda kirjutada. Kuna teema oli väga keeruline ja suur, võttis algoritmi arendus aset paberlehel.

Alustuseks sai joonistatud üks näiteneuronvõrk, mille alusel sai programmi teha. Joonisel tuli ära märkida iga väiksem detail, kirjutada selgelt lahti erinevate komponentide ehitus ja eesmärk, ning tuua välja joonisel kujutatud võrgustiku eeldatav töökäik, mille järgi sai hiljem programmi tööd kontrollida.

Kuigi masinõppe kontsept oli pigem lihtne ja mõistetav, siis paberil sellise projekti kujundamine ja arendamine osutus palju keerulisemaks, kui arvata võis. Kõige tähtsam algoritmi arendamise juures paberil oli see, et ise tehtud skeemid hiljem lihtsad ja arusaadavad oleksid.

Igat väiksemat funktsiooni, ideed ja arvutuskäiku, mida paberile kanda sai, sai ka kontrollitud koheselt arvutis, kasutades keelt Python. Kontrollida tuli põhjalikult, sest väikseimgi viga võis tähendada algoritmi mittetöötamist ja pikka ning tüütut veaparandamist.

Terve töö kõige raskemaks etapiks osutus just arendus. Ühe varasematest valmis skeemidest sai valmis umbes kuu peale arenduse algust. Skeeme sai palju täiendatud ka peale valmimist.

### **2.3 Keele valimine ja õppimine**

Peale esialgsete skeemide valmis saamist tuli kõige tähtsam otsus, milleks oli programmeerimiskeele valimine. Python oli küll lihtne ja otsekohene ning terve arenduskäigu jooksul oli kasutusel just see keel, kuid siiski oli Pythonil mõningad tähtsad puudujäägid.

Keelele, millele algoritm töötama pidi, sai projekti raames kehtestatud mõned nõuded. Nendest kõige tähtsamad nõuded olid keele jõudlus, keele ehitus ja lihtsus ehk õpitavus. Peale selle oli ka soov, et keelel oleks suur kasutajaskond, sest sellisel juhul on keelt lihtsam õppida ja foorumites on rohkem inimesi, kes häda korral aidata saaksid.

Jõudlus on masinõppe juures ülimalt tähtis, sest masinõppe algoritm eeldab miljoneid arvutusi ja ülesannete täitmist. Iga väiksemgi kiiruse erinevus võis pikas perspektiivis tähendada, suuremate neuronvõrede puhul, väga pikka aega. Enamasti tähendab kiirem keel raskemini õpitavust, seega tuli valida keel, millel on sarnane ehitus varasemalt omandatud keeltega.

Peale erinevate keelte võrdlemist langes otsus kasutada programmeerimiskeelt Java. Java on maailmas üks kõige populaarsemaid keeli4, millele annab juurde ka see, et keel on ametlik operatsioonisüsteem Androidi keel. Populaarsus tähendas seda, et keelt arendatakse aktiivselt ja selle kasutamise kohta võib leida internetis palju abi. Keele üks loojatest märkis viis põhimõtet, mida keel peab järgima, millest kaks olid lihtsus ja jõudlus5 - täpselt see, mida projekti jaoks vaja.

Javat sai õppida paljudes tuntud veebipõhistes õppekeskkondades. Olles varemgi programmeerimist internetis õppinud, oli autoril välja kujunenud lemmik lehekülg, kus keelt õppida sai. Leheküljeks oli codecademy, mis oli täis õpetavaid pilte, interaktiivseid harjutusi, elupõhiseid ülesandeid ja ka meeldivat huumorit. Leheküljel oleva Java kiirkursus kestis nädala või kaks, enamasti tegeledes sellega peale kooli kodus.

### **2.4 Programmeerimine**

Kõige viimane osa projekti juures oli selle projekti programmeerimine. Esimesed päevad käis programmeerimine programmis Notepad++, sest sellega oli juba varasemaid kogemusi ning oli teada, et see toetab ka Javat. Kuigi toetas Notepad++ Javat, tekkis sellega vahel probleeme ja foorumitest sai teada palju paremast alternatiivist, milleks oli IntelliJ IDEA. Enamuse ajast kasutasin seega programmeerimiseks keskkonda IntelliJ IDEA Community Edition väljaannet 2017.3.4.

Kui siiamaani tehtud programmidest on enamus muutunud lõpupoole kaootilisteks ja koodist pole enam midagi aru saada, siis selle projekti juures tuli veenduda iga uue rea kirjutamisega, et kõik oleks organiseeritud ja loetav. Õnneks olid skeemid juba tehtud ja järele jäi ainult kood kirjutada, seega korrastatuse hoidmine oli lihtsam. Juba ette planeeritud programmi kirjutamine, oli lihtsam ja meeldivam kui kohapeal väljamõeldu kirjutamine.

Erinevused Javas ja Python’is on suured, suuremad, kui alguses arvatud sai. Sellest tulenes ka terve projekti üks ainsatest ebameeldivatest probleemidest. Nimelt kuna skeemide tegemisel oli kasutusel Python, ja programmi ise sai kirjutada Javas, tuli programmeerides skeeme nö tõlkida ja väike osa skeemist tuli ka ümber teha. Peale nende ebamugavuste, läks programmi programmeerimine hõlpsasti.

Tulemuseks sai hästi dokumenteeritud, organiseeritud ja paindliku programmi, mille õpetamine ja modifitseerimine on seletatud ja hõlbus. Algoritmi pikkuseks kokku tuli umbes 500 rida, kui sisse arvestada kommentaare ja tühju ridu, millega programm vormistatud on.

# 

## **3. TULEMUSED**

### **3.1 Programmi testimine**

Esimeste silmaga nähtavate tulemusteni jõudmiseks kulus kuid õppimist ja arendamist. Esimesed väikesed testid sai teha tegelikult juba programmeerimise käigus, kuid lõpuks sai panna algoritm tõeliselt proovile.

Esimesed testid olid tagasihoidlikud. Testimiseks said loodud väga lihtne ülesanne, milleks oli kahte juhuslikult genereeritud numbrit võrrelda, kas esimene on teisest suurem või mitte. Programm õppis tsüklite kaupa, mida antud juhul oli umbes paartuhat. Kuna neuronvõre oli väike, ülesanne lihtne ja tsükleid tuli teha vähe, lõpetas algoritm õppimise peaaegu momentselt peale alustamist. Tulemusi vaadates, õppis programm kahte arvu võrdlema umbes 90% täpsusega.

Soovides paremat tulemust, sai panna programm õppima pikemalt, kui eelmine kord. Ka seekord lõpetas programm väga kiiresti, kuid seekord olid tulemused paremad. Programm oskas kahte arvu võrrelda õigesti rohkem, kui 99% kordadest. Kuid tulemusi lähemalt uurides, avastasin, et programm jõudis sellise tulemuseni palju varem ja kuskil tuli ette arengu lagi.

Tulemusi sai proovida parandada erinevalt, kuid arengu lagi ei kadunud, parimal juhul nihkus tulemus ainult natukene kõrgemale. Seega sai jätta numbrite võrdlemine ja hakata testima programmi millegi muuga.

Järgmised testid olid mängude põhilised. Mänguks sai valitud Pong. Algoritmi ümberseadistamine oli väga lihtne, kuna arenduse käigus sai sellele juba pikalt ette mõelda. Muudatuste tegemiseks oli vaja ainult mõned numbrid ära muuta ja anda algoritmile koordinaadid.

Pong’i puhul olid sisenditeks palli ja mängja ordinaadid ja väljunditeks kolm väärtust: üles, alla või neutraalne, mis olid siis mängja liikumisjuhendid. Õppimise ajaks tuli mäng teha kiiremaks, sest muidu oleks läinud algoritmil selle õppimisega liiga kaua aega. Tulemuseks sai väga osava tehisintellekti, mis ei saanud pallile pihta ainult väga äärmistel juhtudel.

# 

### **3.2 Praktilise töö tulemused**

Praktilise töö tulemusena on valminud nullist tehtud programm, mis suudab ise õppida erinevate ülesannete lahendamist, mida on see ka vähemalt kahel korral tõestanud. On valminud algoritm, mis on õppinud võrdlema kahte arvu ja mängima Pong’i. Praktilise tööga alustades, oli küll lootus teha iseõppiv tehisintellekt, kuid ei osanud lootagi teha selline, mis õpib mängima videomängu

Terve töö raames on omandanud autor uusi teadmisi masinõppest, õppinud ise tegema masinõppe algoritmi, valdanud uue programmeerimiskeele ja parendanud programmide arendusoskust. Eesmärgiks oli luua, õpetada ja panna edukalt proovile tehisintellekt, ilma väliste ettekirjutatud programmidega, ja nendega saadi edukalt hakkama. Tuli välja isegi rohkem, kui eesmärgiks seatud oli.

Suurim puudujääk töö juures oli algoritmi viimistlus. Programmi oleks arvatavasti saanud teda märkimisväärselt effektiivsemaks ja tõsta selle oskuslage, ku programm ülesannete arenemisega kinni jäi. Üheks miinuseks oli ka oskamatus erinevate ülesannetelt saadud andmeid analüüsida.

Tööga tegelemine insipreeris autorit ka tulevikus masinõppe kohaseid projekte looma. Tänu tööle saadud kogemustega, läheb kindlasti ka tulevaste algoritmide koostamine hõlpsamini ja tulemused tulevad paremad.

## **KOKKUVÕTE**

Eesmärgiks oli seatud luua iseõppiv tehisintellekt, mis suudaks õppida erinevate ülesannete lahendamist ja ka lahendada neid ülesandeid, ning selle loomise käigus omandada ja õppida masinõppe põhimõtteid. Programmi testides on näha, et algoritmi põhilised eesmärgid on täidetud. Programm on tänaseks tõestanud, et suudab õppida numbrite võrdlemist ja videomängu Pong.

Lisaks tehisintellekti valmistamisele, õppis autor ära ka uue programmeerimis keele ja leidis endale uue keskkonna, IntelliJ, kus saab programme kirjutada. Varasemalt ei olnud autor teinud mitte ühtegi tehisintellekti, eriti sellist, mis ise õppida suudab.

Projekti tegemisega läks tegelikut väga kaua. Alustuseks tuli selgeks teha, kuidas ültse masinõpe toimib. Selle jaoks oli vaja leida internetist õppekursus ja see läbida, mis sai ka tehtud. Isegi peale kursuse läbimist õppis autor sageli projektiga tegeledes midagi uut juurde.

Oluline punkt on see, et ma sai iseseisvalt arendada programm algusest lõpuni, ilma et oleks väliseid programmeerimis raamistikke kasutatud. Neid kasutades oleks autor jäänud paljust ilma ja ei saaks algoritmi täiesti enda omaks kutsuda.

Mis autor sellest tööst veel kaasa saab, on uus keel Java, mis on kujunenud ka autori uueks lemmikuks keeleks. Olles kasutanud keelt põhjalikult, avastas autor, et ka tulevikus oleks mõtekam ja meeldivam just Javat kasutada. Keelega oli meeldiv programmeerida ja tulemus oli üllatavalt hea.

Programmi sai testitud ka videomängus ja sellega hakkama saamine on tunnistus algoritmi õppeoskusele. Mõned vead tehisintellektis esinesid ja saadud andmetega ei osatud midagi peale hakata, kuid autor usub, et töö on motiveerinud teda ka edaspidi masinõppega tegelema.

## **KASUTATUD KIRJANDUS**

1. Google Trends, <https://trends.google.com/trends/explore?date=today%205-y&q=machine%20learning> 23.05.2018
2. **Mitchell, T.** (1997). Machine Learning, McGraw Hill
3. **Raimond J. Mooney**, CS 391L: Machine learning introduction
4. **Jamie McKane,** “The most popular programming languages in the world”, <https://mybroadband.co.za/news/software/251609-the-most-popular-programming-languages-in-the-world-8.html> 24.05.2018
5. Sun Microsystems, Inc. Oracle, <http://www.oracle.com/technetwork/java/intro-141325.html> 24.05.2018

## **SELF-LEARNING ARTIFICIAL INTELLIGENCE**

Less than 60 years ago, the idea of machines learning, was a concept you could only find in fiction. We have reached a point in time where machine learning is a part of most peoples’ everyday life. There are even machine learning algorithms that tell you to learn how machine-learning algorithms work. While machine learning is a globally popular subject and can be found almost anywhere you look, it’s still thought of as something very complicated and unreachable by the general public. These are the reasons for which the author decided to study the concept behind those algorithms and create a self-learning program himself.

This paper describes the process of studying machine learning and then creating a program that utilises that. It does so by going through some important points step-by-step.

The very first step of creating a self-learning system, is to understand how they work. For that, a web-based learning environment Udacity, was chosen. The course covers the topic very thoroughly and gives the user a complete overview. Even though some of the parts of this course were skipped, everything needed to create a self learning algorithm was learnt.

The second step of this project was the actual development and planning of this program. Although the concept of machine learning is rather easy and understandable, actually creating it turns out to be truly tough. Being the hardest part of the work, it took a few months to actually create the first complete version. Even after finishing development, were some notable changes made to it. Most of the planning and schemes were made on paper, rather than a computer to keep everything clean and understandable.

Eventually it was time to pick a programming language to write the program in. There was a variety of languages to pick from, but the author had also set a criteria for the language to meet. It had to be fast, due to the nature of machine learning, and easily learnable. Java was chosen as the language, as it made especially to be fast and easy to learn.

Programming itself wasn’t at all difficult, since all the hard work had already been done. Most of the program was written in a professional IDEA called IntelliJ, which made writing rather enjoyable. The only barrier here was translation, since different programming languages were used for development and actually creating the program, and some parts of the scheme, that worked with Python, didn’t with Java.

Lastly, the algorithm had to be tested. It was tested using two different methods. One of them was a simple number comparison and the other a game on pong. Eventually the algorithm had proven it could learn to compare two numbers with a greater than 99% accuracy and play the game pong almost perfectly, rarely having the opponent gain a point.

Considering all of the above, this project was a successful one. The objective of creating a self learning artificial intelligence was completed, resulting in an even better program than anticipated. Moreover, this experience gave the author a good overview of machine learning, taught him a new language and horned his development skills. Although the project was time consuming and at points exhausting, the author belives that he will do more similar projects in the future.