



Manuaalisen todennäköisyysarvion kehittäminen sovellukseksi – Case: Jääkiekko

Johannes Jokinen

Opinnäytetyö, AMK

Toukokuu 2023

Tietojenkäsittelyn tutkinto-ohjelma



Jyväskylän ammattikorkeakoulu
University of Applied Sciences

Jokinen, Johannes

Manuaalisen todennäköisyysarvion kehittäminen sovellukseksi – Case: Jääkiekko

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2023, 35 sivua

Liiketalouden ala. Tietojenkäsittelyn tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: Suomi

Julkaisulupa avoimessa verkossa: ei

Tiivistelmä

Tutkimuksessa kehitettiin sovellusta manuaalisesta menetelmästä laskea jääkiekko-otteluiden todennäköisyyksiä. Alkuperäinen menetelmä vei paljon aikaa, oli monimutkainen sekä altis virheille datan käsittelyn aikana.

Tavoitteena oli luoda käyttöliittymä, joka laskee automaattisesti todennäköisyydet käyttäjän valitsemaan NHL-otteluun. Lisäksi tavoiteltiin käyttöliittymää, joka on mahdollisimman yksinkertainen, jotta mahdollinen uusi käyttäjäkin voi toteuttaa arvion tekemisen ilman erityisiä ohjeita. Sovellusta lähdettiin kehittämään, koska vanha menetelmä oli erityisen hidas, joten myös arvion tuottaminen mahdollisimman nopeasti oli tärkeä tavoite.

Sovellus kehitettiin toimimaan pääosin Python -kielellä käyttäen apuna Flask -ohjelmistokehystä. Datan käsittely automaattisesti toteutettiin Pandas -kirjastolla, joka on luotu datankäsittelyä varten.

Lopputuloksena syntyi sovellus, joka on äärimmäisen helppokäyttöinen, nopea ja poistaa kokonaan käyttäjän tekemät virheet dataa käsiteltäessä. Alkuperäisellä menetelmällä yksittäisen ottelun arvioimiseen kului aikaa hieman päälle kymmenen minuuttia. Tutkimuksessa luotu sovellus käyttää täsmälleen saman arvion tuottamiseen noin kaksikymmentä sekuntia.

Avainsanat (asiasanat)

Kehittämistutkimus, sovelluskehitys, Python, todennäköisyysarvio

Muut tiedot (salassa pidettävät liitteet)

-

Jokinen, Johannes

Developing manual probability estimation into an application

Jyväskylä: JAMK University of Applied Sciences, May 2023, 35 pages

Business Information Technology. Degree programme in Business Information Technology. Bachelor's thesis.

Permission for open access publication: No

Language of publication: Finnish

Abstract

In a study an application was developed to automate the process of calculating probabilities for ice hockey matches, which was originally a time-consuming, complex and error-prone manual method.

The goal was to create a user interface that automatically calculates the probabilities for the user's selected NHL match and is as simple as possible so that even a new user can make an estimate without special introduction. The application was developed because the old method was particularly slow, so producing the estimate as quickly as possible was also an important goal.

The application was mainly developed using the Python language and the Flask framework. The data processing was automatically implemented using the Pandas library, which is designed for data processing.

The end result was an extremely easy to use, fast application that completely eliminates user errors when processing data. The original method took just over ten minutes to evaluate a single match, while the application developed in the study takes about twenty seconds to produce exactly the same evaluation

Keywords/tags (subjects)

Development research, application development, Python, probability estimation

Miscellaneous (Confidential information)

-

Sisältö

1	Johdanto	6
2	Tutkimusasetelma	8
2.1	Opinnäytetyön tarkoitus ja tavoitteet	8
2.2	Käytetyt tutkimusmenetelmät	9
2.3	Tutkimusongelma- ja kysymykset	9
3	Todennäköisyysslaskenta	11
3.1	Todennäköisyysslaskenta yleisesti.....	11
3.2	Todennäköisyyksien laskeminen urheilutapahtumasta.....	12
4	Prosessissa käsiteltävät teknologiat	15
4.1	Datan käsittely prosessin aikana	15
4.2	Käyttöliittymä sovellukselle	18
4.3	Käyttöliittymää varten tutkittavat teknologiat	20
5	Todennäköisyysslaskennan kehittäminen	22
5.1	Prosessin tilanne alkuvaiheessa	22
5.2	Prosessin kehittäminen sovellukseksi	23
6	Tutkimustulokset ja pohdinta	31
6.1	Käyttöliittymän toteutus	31
6.2	Datan käsittely Pythonin avulla.....	32
6.3	Kehittämistyön tulokset	32
6.4	Opinnäytetyön eettisyys ja luotettavuus	33
6.5	Jatkotoimenpiteet	33
Lähteet		34

Kuviot

Kuvio 1. Esimerkkejä eri joukkueiden voimaluvuista.....	13
Kuvio 2. Yksinkertainen malli maaliodottamasta.	14
Kuvio 3. Kuvassa on taulukko yksittäisen joukkueen pelaajatilastoista	16
Kuvio 4. Simulaation tulostama todennäköisyysarvio	23
Kuvio 5. Esimerkkikuva Flaskin käytöstä.....	25
Kuvio 6. Esimerkkikuva Pandas -kirjaston käytöstä.	26
Kuvio 7. Esimerkkikuva HTML -tiedostosta.....	27
Kuvio 8. Esimerkkikuva Python -tiedostosta.....	28
Kuvio 9. Kuva käyttöliittymästä 1.....	29

Kuvio 10. Kuva käyttöliittymästä 2	29
Kuvio 11. Kuva käyttöliittymästä 3	30

1 Johdanto

Todennäköisyysarvio on luku, jolla kuvataan minkä tahansa tapahtuman todennäköisyyttä. Kyseessä ei siis ole kyseisen tapahtuman tarkka todennäköisyys, vaan arvio, joka on tehty käytännössä millä tahansa tekniikalla. Kun katsotaan todennäköisyysarviota jääkiekko-ottelusta, on tapoja tuottaa todennäköisyysarvio loputtomasti. Todennäköisyysarvion voi tehdä esimerkiksi vain katsomalla joukkueiden pelejä ja arvioimalla silmämääräisesti kumpi joukkue on parempi. Arvioita voidaan tehdä myös katselemalla edellisten pelien tuloksista, miten kyseiset joukkueet ovat pelanneet, tai kuinka ne sijoittuvat sarjataulukossa. Edellä mainitut toimintatavat ovat kuitenkin erittäin vahvasti riippuvaisia arvion tekijän tasosta tulkita pelitapahtumia tai -tuloksia oikein, eikä arvio välttämättä ole lähellä todellisuutta. Mitä tarkemman arvion ottelutapahtumasta haluaa tehdä, sitä enemmän on osattava hakea ja käsitellä dataa siitä ketkä ottelussa pelaavat ja miten he ovat aikaisemmin suorittaneet.

Tämä opinnäytetyö ei ota kantaa siihen, tai tutki sitä, mikä olisi paras tapa tuottaa todennäköisyysarvio jääkiekko-otteluun. Tarkoituksena on käyttää jo olemassa olevaa manuaalisesti tapahtuvaa todennäköisyyksien laskentatapaa, jonka alusta loppuun suorittaminen voi kestää nykyisellä toteutustavalla otteluiden määrästä riippuen useita tunteja – ja muokata siitä sovellus, jolla saa tuotettua saman lopputuloksen korkeintaan muutaman minuutin työnteolla. Opinnäytetyössä tutkitaan erilaisia vaihtoehtoja tuotoksen toteuttamiselle ja sen jälkeen valitaan tutkimuksen kohteena olleista vaihtoehdoista paras.

Kyseessä on siis kehittämistutkimus, jonka tavoitteena on luoda huomattavasti nopeampi tapa tuottaa todennäköisyysarvio samalla määrällä datankäsittelyä, kuin mitä aiemmassa prosessissa on käytetty. Tämä mahdollistaa kyseisen prosessin tekemisen nopeasti päivittäin sen sijaan, että siihen joutuisi käyttämään useita tunteja joka kerta. Kun asiaa katsotaan todennäköisyysarvion tuottamisen näkökulmasta, niin päivittämällä datan ennen jokaista arviota saa huomattavasti tarkemman arvion tehtyä, jos vertaa siihen, että data on päivitetty viimeksi esimerkiksi kuukausi sitten.

Opinnäytetyössä tutkitaan uusimpia teknologioita ja vertaillaan niiden hyviä ja huonoja puolia opinnäytetyössä kehitettävän prosessin suhteen. Pääosin tutkimusta tehdään Python-kieleen liittyvien vaihtoehtojen suhteen, koska olemassa olevat menetelmät, sekä simulaatio on rakennettu

sen päälle. Kun vaihtoehdot ovat selvillä, kehitetään toimiva sovellus, jonka avulla prosessin saa suoritettua päivitetyllä datalla huomattavasti nopeammin kuin aiemmin.

Opinnäytetyön aloituspisteessä todennäköisyyksien laskeminen on erittäin sekavaa ja suhteellisen virheeltistä. Tilastot lähes tuhannesta pelaajasta haetaan Exceliin, jonka jälkeen noin 950 pelaajaa jaetaan omien joukkueidensa välilehtiin. Tilastoja ei suodateta millään tapaa tässä vaiheessa, joten taulukkoihin tulee todella paljon myös epäolennaista dataa. Kun kaikki asiat käsitellään manuaalisesti, niin kaikki ylimääräinen data, joka näyttää eri rivien välillä hyvin samanlaiselta nostaa virheen tapahtumisen todennäköisyyttä huomattavasti. Varsinkin joukkuekohtaisia voimalukuja laskeessa yksittäinen virhe voi vaikuttaa huomattavasti voimalukuihin, mikä taas vaikuttaa lopulliseen arvioon. Näistä syistä menetelmän automatisointi sekä nopeuttaa sen alusta loppuun viemistä, mutta myös poistaa manuaaliset virheet koko prosessista.

2 Tutkimusasetelma

2.1 Opinnäytetyön tarkoitus ja tavoitteet

Tämän opinnäytetyön tarkoituksena on tutkia erilaisia teknologioita sekä viitekehyksiä ja niiden soveltuvuutta todennäköisyysarvion tuottamiseen tarkoitettun sovelluksen luomiseksi. Opinnäytetyön aihe on valittu sen takia, koska tällä hetkellä ei julkisesti ole saatavilla urheilutapahtumiin tilastojen avulla todennäköisyysarviota laskevia sovelluksia, joten tähän kysyntään tuotetulla sovelluksella pyritään vastaamaan.

Opinnäytetyön aloituspisteessä käytetyt menetelmät todennäköisyyksien laskemiseen jääkiekko-otteluista ovat erittäin hitaat. Kyseisellä menetelmällä yhden ottelun arvion tuottamiseen menee reilusti yli kymmenen minuuttia. Jos otetaan esimerkiksi NHL ja lasketaan kuinka paljon aikaa yhden kierroksen otteluiden todennäköisyyksiin laskemiseen menee aikaa, niin tulokset ovat seuraavanlaisia; Joukkueita on 32. Tämä tarkoittaa sitä, että yhdessä päivässä voi olla maksimissaan 16 ottelua. Kun jokaisen ottelun todennäköisyyksien laskemiseen käytetään erittäin optimistisella arviolla esimerkiksi 10 minuuttia, saadaan aika-arvioksi 160 minuuttia, joka tarkoittaa sitä, että yhtä kierrosta varten aikaa kuluisi arviolta kaksi tuntia ja 40 minuuttia. Kun otetaan huomioon, että sarjassa pelataan 82 kierrosta, niin se tarkoittaa sitä, että otteluita on yhteensä 1312. Jos laskennallisesti mietitään, kuinka monta tuntia tämä vie koko kauden aikana, tulee tunteja lähes 220. Mikäli laskenta hoituisi sovelluksen avulla automatisoidusti niin, että käyttäjän tulisi vain valita ottelussa pelaavat joukkueet ja laskea arviot, niin tämä säästäisi huomattavasti aikaa. Jos lasketaan esimerkiksi, että sovelluksen avulla yksittäisen ottelun arvion aika saataisiin kahteen minuuttiin, tämä vähentäisi lähes 180 tuntia vuodessa. Opinnäytetyön aloituspisteessä olevia menetelmiä kuvataan tarkemmin luvussa 4.1

Tavoitteena on siis luoda sovellus, jota käyttämällä henkilö säästää aikaa opinnäytetyön alkupisteessä olevaan käytettyihin menetelmiin nähden ja saa päivitettyä kyseiset menetelmät käyttäen mahdollisimman samoilla toimintatavoilla toteutetun lopputuloksen tutkimastaan jääkiekko-ottelusta mahdollisimman nopeasti.

Tutkimukseen valittu kehitettävä prosessi on opinnäytetyön aloitushetkellä vaikea toteuttaa, koska se on erittäin monimutkainen ja se vie paljon aikaa. Kehittämisen tarkoituksena on luoda sovellus, jota osaa käyttää sellainenkin henkilö, joka ei tiedä aiheesta mitään. Optimaalisessa tilanteessa lopputuotos on niin selkeä ja yksinkertainen, että asiasta mitään tietämätön henkilökin saa prosessin suoritettua alusta loppuun.

2.2 Käytetyt tutkimusmenetelmät

Opinnäytetyön tietopohjaa varten kerättiin tietoa kirjallisista lähteistä sekä verkkojulkaisuista. Kirjallisia lähteitä käytettiin pääasiassa opinnäytetyön tuottamisen apuna, jotta opinnäytetyö pystyttiin tuottamaan laadukkaasti noudattaen hyviä tutkimusperiaatteita. Opinnäytetyön kehittämistutkimuksen suorittamista varten haettiin tietoa ainoastaan verkkojulkaisuista- ja artikkeleista. Kehittämiseen liittyvä tiedonhaku oli laadukkaampaa etsiä verkosta kirjallisten lähteiden sijaan, koska ala on koko ajan muuttuva ja uutta tietoa uusista teknologioista ilmestyy verkkoon joka päivä. Pääasiassa hakutermit liittyivät ohjelmistokehyksiin, sekä datankäsittelyssä käytettäviin työkaluihin, kuten esimerkiksi ”Pandas” ja ”Flask”

Opinnäytetyö toteutettiin kehittämistutkimuksena. Kehittämistutkimus on termi jota käytetään, kun jotain asiaa halutaan kehittää ja tutkia ongelmaa syklisessä prosessissa. Kehittämistutkimuksen päämääränä on tuottaa käytännössä toimivia ratkaisuja, sekä muutosta lähtötilanteeseen. Jotta opinnäytetyötä voi kutsua kehittämistutkimukseksi, on siinä oltava tutkimus ja tulosten sekä tutkimusprosessien raportointi. (Kananen, 2015.)

2.3 Tutkimusongelma- ja kysymykset

Opinnäytetyössä käsiteltävä tutkimus liittyy todennäköisyyksien laskemiseen jääkiekko-otteluista. Aloituspisteessä käytössä oleva menetelmä on hidas, sekä altis virheille. Tämän lisäksi se on erittäin monimutkainen. Menetelmää kuvataan tarkemmin luvussa 4.1.

Yllä mainituista syistä johtuen tutkimuksen haasteina on löytää keinot siihen, miten saadaan samat lopputulokset käyttäen nopeampaa menetelmää, joka myös poistaa manuaalisesta työstä johtuvat virheet mahdollisimman laajasti. Opinnäytetyön tavoitteena oli luoda käyttöliittymä menetelmälle, joka laskee todennäköisyydet samoilla kaavoilla alkuperäisen menetelmän kanssa, mutta

toteuttaa samat vaiheet automaattisesti. Kyseinen ratkaisu mahdollistaa sekä ajan säästämisen, että myös virheiden vähentämisen. Näistä syistä tutkimuskysymykset olivat seuraavanlaiset:

Kuinka toteuttaa käyttöliittymä, josta käyttäjä voi arvioida tiettyjen joukkueiden voimasuhteita?

Kuinka automatisoida datankäsittely mahdollisimman pitkälle?

3 Todennäköisyyslaskenta

3.1 Todennäköisyyslaskenta yleisesti

Todennäköisyyslaskenta on matematiikkaan liittyvä termi, joka tarkoittaa sitä, että pyritään ennustamaan tulevien tapahtumien todennäköisyyttä. Todennäköisyyttä tapahtumalle kuvataan usein prosenteilla (0-100). Mitä pienempi numero jonkin tapahtuman todennäköisyydelle arvioidaan, niin sitä harvemmin se tapahtuu ja vastaavasti jos arvio on 100%, se tulee tapahtumaan täysin varmasti. (Yle 2013.)

Tapahtuma

Tapahtuma on joukko alkeistapauksia, joille on mahdollista laskea todennäköisyys. Kun puhutaan kaikkien alkeistapauksien joukosta, siitä käytetään termiä perusjoukko. Kun halutaan viitata vain tiettyyn osaan perusjoukosta, käytetään termiä osajoukko. Osajoukko on esimerkiksi korttipakasta kaikki ruudut, kuvakortit tai parittomat numerokortit. (Gröhn, J., Huusko, J-M., Juvonen, M. & Pirhonen, E. n.d.).

Alkeistapaus

Alkeistapauksella tarkoitetaan satunnaisilmiön mahdollista lopputulosta. Esimerkiksi kolikkoa heittäessä kruuna ja klaava ovat molemmat alkeistapauksia. Ja korttipakasta sattumanvaraisesti nostettu kortti on yksi 52:sta alkeistapauksesta. (Gröhn, J. ym. n.d.)

Todennäköisyysjakauma

Todennäköisyysjakauma on luku, joka kertoo todennäköisyyden jokaiselle mahdolliselle lopputulokselle. Riippumatta siitä, kuinka paljon mahdollisia vaihtoehtoja lopputulokselle on, todennäköisyysjakauman jokaisen vaihtoehdon yhteenlaskettu summa on aina 1. Todennäköisyysjakauma voi jakautua loputtomalle määrälle vaihtoehtoja ja jakaumat voivat olla samaa, tai eri arvoa muiden vaihtoehtojen kanssa. (Yle 2013.)

Klassinen todennäköisyys

Klassinen todennäköisyys tarkoittaa tapahtumaa, jolla on tietyt tunnusmerkit. Näitä tunnusmerkkejä ovat tapahtuman satunnaisuus, se, että tapahtuman jokaisella lopputuloksella on yhtä iso mahdollisuus toteutua, tulos on jokin tiedossa olevista ja ennalta määritellyistä vaihtoehtoista ja että vain yksi lopputuloksista toteutuu. Klassisesta todennäköisyydestä hyviä esimerkkejä ovat esimerkiksi kolikon, tai nopan heitto. Molemmissa tapahtumissa on ennalta tiedossa kaikki mahdolliset lopputulokset ja jokaisen lopputuloksen todennäköisyys on yhtä suuri. (Gröhn, J. ym. n.d.)

Komplementti

Komplementilla tarkoitetaan kaikkia muita lopputuloksia, kuin tapahtunutta lopputulosta. Esimerkiksi jos heitetään noppaa ja lopputulokseksi tulee luku 4, niin siinä tapauksessa kaikki muut vaihtoehdot, jotka eivät toteutuneet ovat komplementteja. (Gröhn, J. ym. n.d.)

3.2 Todennäköisyyksien laskeminen urheilutapahtumasta

On olemassa lukemattomia erilaisia urheilutapahtumia. Jokaiseen erilaiseen lajiin löytyy myös erilainen tapa arvioida todennäköisyyksiä. Kun yritetään tuottaa mahdollisimman tarkkaa arviota tietyistä tapahtumista, niin eri urheilulajeissa on otettava erilaisia muuttujia huomioon. Lajista riippuen muuttujia voi olla vain muutamasta todella moniin.

Jos verrataan arvioitavien muuttujien määrää esimerkiksi tenniksen (kaksinpeli) ja jääkiekon välillä, löydetään paljon isoja eroja. Kun tarkastellaan tennistä, niin arviota tehdessä on vain muutama muuttuja. Säännöt ovat myös selkeät ja tuomarilla on mahdollisuus tarkastaa tapahtumat videolta, joten sattuman osuus on hyvin pieni.

Jääkiekossa taas molemmilla joukkueilla on kaksikymmentä pelaajaa, jotka voivat vaihtua pelistä riippuen. Muuttujia jääkiekko-ottelussa on lukematon määrä, johtuen isosta määrästä pelaajia ja vastapuolen pelaajia, jotka sattuvat samaan aikaan kentälle. Tuomareita tarvitaan neljä kappaletta kentän puolelle ja isossa osassa tilanteista ei ole mahdollista tarkistaa tapahtumia videolta. Lisäksi sääntöjen tulkitseminen tapahtuu jokaisen tuomarin pienessä ajassa tekemällä päätöksellä.

Voimaluku

Voimaluku on luku, jolla pyritään kuvaamaan joukkueiden välisiä voimasuhteita toisiinsa nähden. Urheilutapahtuman todennäköisyyttä arvioidessa voimalukuja käytetään arvioimaan joukkueen suhdetta toisiinsa samalla periaatteella, kuin sarjataulukkoa katsomalla, mutta sillä erotuksella, että voimaluku antaa tarkemman kuvan joukkueiden todellisesta tasosta, kuin sarjataulukko. Sarjataulukko ottaa huomioon vain menneiden otteluiden tulokset, kun taas voimalukua laskiessa pyritään laskemaan joukkueen taso pelitapahtumien ja tilastojen avulla. Voimaluku voi huomioida esimerkiksi joukkueiden loukkaantumistilannetta, motivaatiota tai kotietua. (Ylikerroin, 2017.)

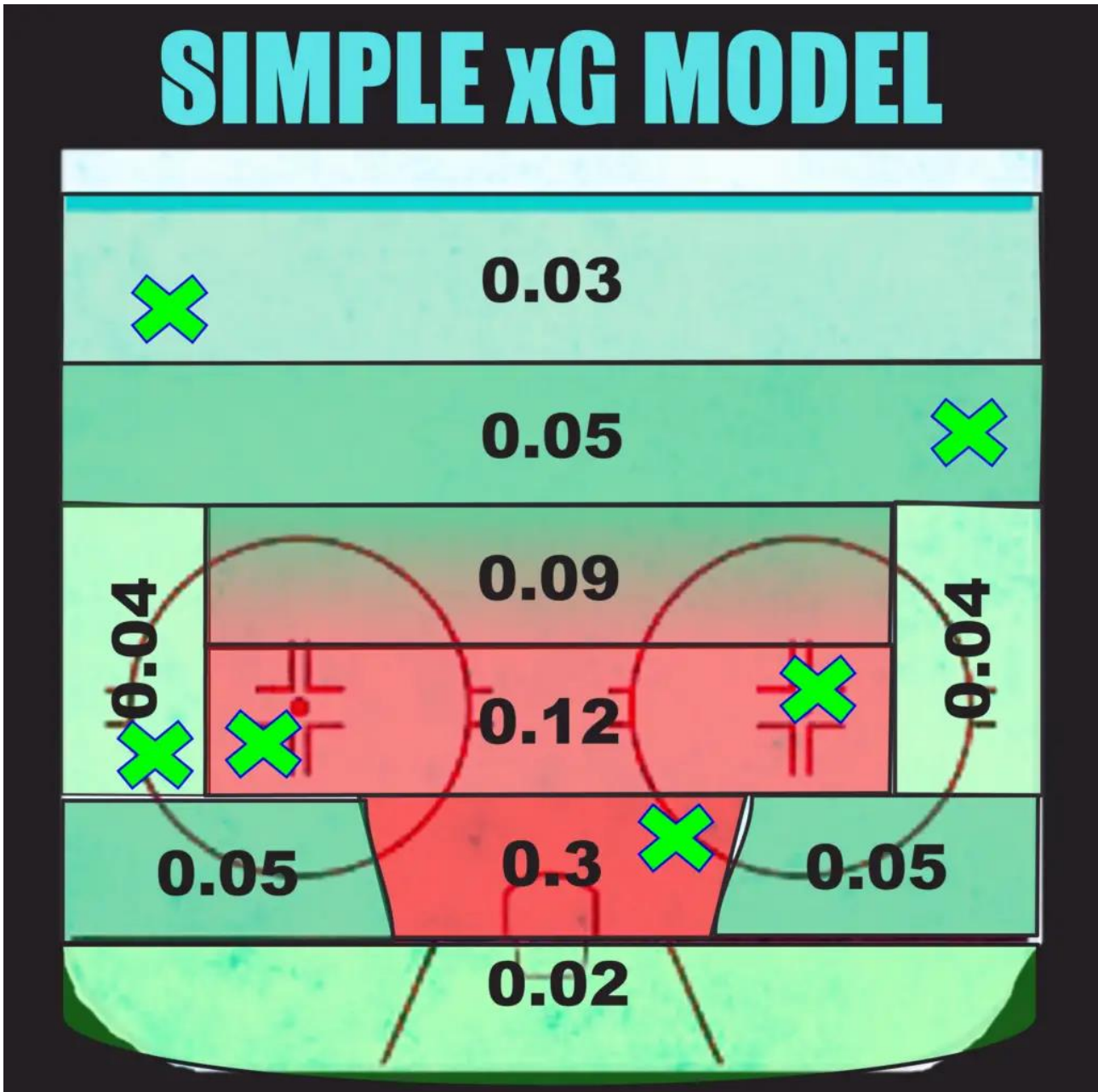
x	TB	CBJ	BOS	PHI	MTL	CGY	NYII	WSH	VGK
tempo	397	357	354	396	410	410	391	419	434
mv	1006	985	1008	1000	1000	1010	1011	1007	1011
xG	56	48	53	50	56	49	52	48	59
shooting	1007	1002	1008	1005	999	1000	1005	1006	1004
PK	998	987	996	991	1003	1002	993	998	1001
nimi	TB	CBJ	BOS	PHI	MTL	CGY	NYII	WSH	VGK
PP	999	991	1009	992	990	997	993	1004	1010

Kuvio 1. Esimerkkejä eri joukkueiden voimaluvuista.

Maaliodottama

Maaliodottama (xG) tarkoittaa yksittäisen maalintekotilanteen todennäköisyyttä sille, että kyseisestä laukauksesta tulee maali. Suurimmat tekijät maaliodottaman laskemiseen ovat laukauksen lähtöpisteen etäisyys ja kulma vastustajan maaliin nähden. Maaliodottaman voi laskea monella eri tavalla riippuen siitä, kuinka tarkan siitä haluaa ja minkälaiset resurssit ovat käytettävissä. Yksinkertaiseen maaliodottaman laskemiseen riittävät vain yllä mainitut tiedot, kun taas tarkempaan laskelmaan voidaan ottaa huomioon esimerkiksi jääkiekossa puolustuksen tai maalivahdin sijoittuminen, tai jalkapallossa se, onko laukaus lähtenyt päällä vai jalalla. Yksittäisen tilanteen maaliodottamasta käytetään lukua nollan ja yhden välillä. Mitä lähempänä luku on ykköistä, sitä todennäköisemmin tilanteesta voi odottaa maalia. (Nordicbet 2020.)

Alla olevassa kuvassa on yksinkertainen malli maaliodottaman laskemiselle jääkiekossa. Vihreät rastit kuvaavat laukauksen lähtöpistettä ja eri väriset alueet kuvaavat laukauksen lähtöpisteen laa-
tua. Mitä isompi lukema kyseisellä värillä on, sitä todennäköisemmin laukauksen voi odottaa menevän maaliin. Maaliodottamaa varten luotuja malleja on loputtomasti. Tässä mallissa ei oteta esi-
merkiksi kantaa siihen, että onko laukausta edeltänyt syöttö esimerkiksi kentän keskilinjan yli, mikä lisää merkittävästi maalin todennäköisyyttä. Mitä suurempi numero on kyseessä, sitä toden-
näköisemmin laukaus menee maaliin.



Kuvio 2. Yksinkertainen malli maaliodottamasta. (Piccolo 2022.)

4 Prosessissa käsiteltävät teknologiat

4.1 Datan käsittely prosessin aikana

Python

Python on vuonna 1991 julkaistu ohjelmointikieli, jota voi käyttää esimerkiksi sovellusten tekemiseen, tehtävien automatisointiin, koneoppimiseen tai data-analyysiin. Python on jo pitkään ollut yksi suosituimmista ohjelmointikielistä, joten siihen on tehty paljon erilaisia kirjastoja kehittämistä varten. Nykyisen prosessin kaikki koodit on tehty pythonilla. (Python Introduction. N.d.)

Opinnäytetyön aloittamishetkellä lähes kaikki datan käsittely tapahtuu excel-sovelluksessa. Prosessin aikana joudutaan siirtämään paljon data paikasta toiseen ja joka paikassa muokkaamaan sitä käsin. Tämä vie ylivoimaisesti eniten aikaa koko prosessissa ja sitä varten on tutkittava vaihtoehtoja siihen, miten datan saisi käsiteltyä mahdollisimman nopeasti. Optimaalisessa tilanteessa käyttäjä voi valita olla muokkaamatta dataa ollenkaan itse. Myös mahdollisuus sen muokkaamiseen on oltava olemassa jokaisessa kohdassa, missä sitä käsitellään. Prosessin optimointia varten on selvitettävä parhaat mahdollisuudet hoitaa jokainen vaihe Python - koodien avulla. Tällä hetkellä data haetaan Exceliin ilman minkäänlaista manipulaatiota. Optimaalisessa tilanteessa käyttäjällä on mahdollisuus Excelin sijaan muokata data suoraan käyttöliittymässä. Tämä tarkoittaa siis sitä, että jo dataa haettaessa, koodin on osattava muokata sitä tarpeeksi pitkälle, valittava oikeat joukkueet, sekä laskettava voimaluvut ennen kuin se näyttää käyttäjälle mitään. Sen jälkeen näitä numeroita pitäisi pystyä muokkaamaan, jonka jälkeen muokatut tai muokkaamattomat numerot menevät arvontakoneeseen, joka näyttää ottelun odotusarvot. Jotta tämä osuus prosessista saadaan optimoitua, on tutkittava mahdollisuudet Python-kirjastoista, joilla saa parhaat mahdolliset tulokset datan käsittelyyn.

Tässä luvussa käydään läpi opinnäytetyössä kehitettävän prosessin datan käsittelyä, sekä siihen liittyviä teknologioita. Prosessin nykytilanteessa kaikki datan käsittelyyn sen hakemisen jälkeen liittyvät asiat tapahtuvat Excelissä ja tämän jälkeen odotusarvot lasketaan käyttäen python- koodia, joka hakee käsiteltyt datat Excelistä. Seuraavissa kappaleissa käydään läpi Excelin ja Pythonin vahvuuksia ja heikkouksia toisiinsa nähden datan käsittelyn suhteen.

	<u>Nimi</u>	<u>Joukkue</u>	<u>Pelipaikka</u>	<u>GP 1</u>	<u>CF</u>	<u>CA</u>	<u>CFp</u>	<u>FF</u>	<u>FA</u>	<u>FF%</u>	<u>SF</u>	<u>SA</u>
9												
10	Roman Josi	NSH	D	69	1462	1276	5340	1088	923	54.1	792	66
11	Craig Smith	NSH	C R	69	847	688	5518	665	516	56.31	479	36
12	Mattias Ekholm	NSH	D	68	1186	1174	5025	918	892	50.72	659	63
13	Ryan Johansen	NSH	C	68	846	805	5124	637	603	51.37	445	43
14	Nick Bonino	NSH	C	67	784	792	4975	615	573	51.77	455	40
15	Matt Duchene	NSH	C	66	937	772	5483	714	591	54.71	507	43
16	Rocco Grimaldi	NSH	C R	66	853	806	5142	665	611	52.12	483	43
17	Calle Jarnkrok	NSH	C	64	766	702	5218	582	530	52.34	419	38
18	Dante Fabbro	NSH	D	64	1042	1019	5056	794	756	51.23	572	53
19	Mikael Granlund	NSH	C	63	887	817	5205	683	612	52.74	482	45
20	Filip Forsberg	NSH	L	63	864	775	5272	641	589	52.11	454	44
21	Kyle Turris	NSH	C	62	751	737	5047	545	565	49.1	397	40
22	Dan Hamhuis	NSH	D	60	647	629	5071	499	464	51.82	344	32
23	Colton Sissons	NSH	C	57	521	621	4562	407	426	48.86	290	30
24	Viktor Arvidsson	NSH	L R	57	718	678	5143	537	514	51.09	365	37
25	Austin Watson	NSH	L	53	413	568	4210	311	399	43.8	214	28
26	Ryan Ellis	NSH	D	49	963	828	5377	723	593	54.94	523	43
27	Yannick Weber	NSH	D	41	508	570	4712	382	427	47.22	253	31
28	Jarred Tinordi	NSH	D	28	338	448	4300	262	340	43.52	184	24
29	Colin Blackwell	NSH	C	27	286	267	5172	216	181	54.41	152	12
30	Yakov Trenin	NSH	C	21	198	202	4950	139	131	51.48	91	8
31	Daniel Carr	NSH	L	11	90	114	4412	67	85	44.08	51	6
32	Mathieu Olivier	NSH	R	8	42	52	4468	34	34	50	30	2
33	Miikka Salomaki	NSH	R	5	35	46	4321	24	29	45.28	19	2
34	Alexandre Carrier	NSH	D	3	40	26	6061	32	21	60.38	23	1
<div><div>◀▶</div><div>TB</div><div>BOS</div><div>VKG</div><div>ARI</div><div>WSH</div><div>PIT</div><div>COL</div><div>NSH</div><div>MTL</div><div>DAL</div><div>FLA</div><div>TOR</div><div>CAR</div><div>STL</div><div>PHI</div><div>CHI</div></div>												

Kuvio 3. Kuvassa on taulukko yksittäisen joukkueen pelaajatilastoista.

Excel ja Python

Datacampin artikkelin (Selvaraj 2023) mukaan, Excel on paras työkalu strukturoidun tiedon käsitte-lyyn, mikä onkin syy siihen, miksi prosessia luodessa Excel on valittu työkaluksi. Toisaalta, kuten Cambridgesparkin (Python vs Excel for Data Analysis 2022) artikkelissa mainitaan, Python pystyy käsittelemään isoja datasettejä paremmin, sekä sen avulla on mahdollista automatisoida Excelissä tapahtuvat tehtävät helposti.

Molemmilla työkaluilla on omat vahvuutensa, sekä heikkoutensa. Excel on loistava työkalu, kun kyseessä on pienet tai normaalit data-analyysiin liittyvät tehtävät, tai esimerkiksi datan visuali-sointiin liittyvät asiat. Excel on myös yksinkertainen ja helppokäyttöinen, vaikka siitä ei olisi aikai-sempaa kokemusta. Yksi Excelin huonoista puolista on se, että sen toiminnot hidastuvat huomatiavasti, mikäli sillä pyrkii tekemään tehtäviä liian isoille dataseteille. (Python vs Excel for Data Ana-lysis 2022.)

Python on Exceliin nähden paljon tehokkaampi ja sillä pystyy käsittelemään myös paljon isompia datasettejä. Pythonilla on myös paljon kirjastoja, joita tulee myös koko ajan lisää, joten Exceliin nähden mahdollisuudet toiminnoille ovat suuremmat, sekä helpommin automatisoitavissa. Pythonin huonoin puoli on se, että sen käyttäminen on paljon vaikeampaa, kuin Excelin. Työtehtävien mahdollisuudet ovat monipuolisemmat, mutta se voi vaatia vuosien kokemusta ohjelmoinnista, jotta niitä osaa käyttää oikein. (Python vs Excel for Data Analysis 2022.)

Olennaiset Python -kirjastot opinnäytetyöprosessissa

Seuraavissa kappaleissa käsitellään pythonin kirjastoja. Opinnäytetyössä käsiteltävässä prosessissa käytetään monia erilaisia kirjastoja, joita on luotu helpottamaan koodin kirjoittamista ja nopeuttamaan erityyppisiä tehtäviä. Luvussa käydään ensiksi läpi, mitä kirjastot ovat, jonka jälkeen katsotaan tarkemmin opinnäytetyön kannalta oleellisia kirjastoja.

Kirjastot yleisesti

Pythonin kirjastoista puhutaan, kun halutaan viitata kokoelmaan esikäännettyjä koodeja. Näitä koodeja voidaan käyttää ohjelmassa tiettyihin tarkasti määriteltuihin toimintoihin. Pythonin kirjastoilla on todella merkittävä rooli, kun ollaan tekemisissä esimerkiksi koneoppimisen, tilastotieteen ja visualisoinnin kanssa. Pythoniin valmiiksi rakennettu ”Python Standard Library” sisältää myös sisäänrakennettuja moduuleja, jotka tarjoavat pääsyn järjestelmän perustoimintoihin. Näihin luokituu yli 200 moduulia, jotka toimivat yhdessä toistensa kanssa, mikä tekee Pythonista korkeatasoisen ohjelmointikielen. (Libraries in Python 2021.)

Pandas

Pandas on erittäin monipuolinen työkalu datan analysointiin pythonin avulla. Kyseessä on kirjasto, jonka avulla voidaan manipuloida, puhdistaa ja analysoida dataa nopeasti ja tehokkaasti. Se mahdollistaa erittäin hyvän tavan työskennellä esimerkiksi taulukoiden kanssa. Se tarjoaa paljon toiminnallisuuksia, kuten tietojen lukemisen CSV-tiedostoista ja taulukoista, sekä näiden esittämistä erilaisilla tavoilla. (Great Learning Team 2022.)

NumPy

NumPy on Python-kirjasto, joka mahdollistaa nopean ja helpon tavan taulukoiden laskemiseen ja prosessointiin. Kirjasto sisältää paljon matemaattisia funktioita, jotka mahdollistavat käyttäjille esimerkiksi vektoroinnin ja matriisitoimintojen hyödyntämisen. Esimerkkeinä voidaan nostaa kirjastosta käytettävistä ominaisuuksista lineaarialgebra ja satunnaislukugenerointi. NumPy tukee myös datan analysoimista ja manipulaatiota mahdollistamalla käyttäjälle taulukoiden helpon luomisen ja manipuloimisen. (What is NumPy n.d.)

4.2 Käyttöliittymä sovellukselle

Käyttöliittymän luominen sovellukselle, jossa käytetään Pythonia ja HTML:llä voi olla haastavaa. Käyttöliittymä on luotava tavalla, joka on käyttäjälle helppokäyttöinen ja tarpeeksi yksinkertainen ymmärrettäväksi. Tämän lisäksi on pidettävä huoli siitä, että Pythonilla luodut koodit ja HTML ovat integroitu oikein.

Ensimmäinen askel käyttöliittymää tehdessä on luoda hyvä suunnitelma siitä, miten sivu asetellaan ja kuinka sitä käytetään. Tärkeää on se, että käyttäjä ymmärtää heti, mitä nappeja tulee painaa ja mitä niistä tapahtuu. Myös napit on laitettava loogiseen järjestykseen. Käyttöliittymän tavoitteena on tehdä sovelluksesta mahdollisimman yksinkertainen ja helppokäyttöinen. Geldartin mukaan parasta on edetä askel kerrallaan aina suunnittelusta testaukseen ja sitä kautta lopputulokseen pääsemiseen. (Geldart 2022.)

Usein, kun luodaan käyttöliittymää, jossa yhdistetään Python ja HTML, käytetään viitekehyskiä, kuten Django tai Flask. PyScript on uusi teknologia näihin verrattuna. Jokainen edellä mainituista teknologioista ja niiden mahdollisuuksista on tutkittava läpi, kun mietitään optimaalista toteutustapaa tämän prosessin muuttamiseksi käyttöliittymän avulla käytettäväksi. Prosessin kannalta on myös erittäin tärkeää, että napeista tapahtuu juuri oikeat asiat datankäsittelyn kannalta, joten jokainen nappi, josta ajetaan koodia, on testattava huolellisesti.

Käyttöliittymää luodessa on siis otettava paljon asioita huomioon, sekä tehtävä tutkimusta siitä, mikä on optimaalinen tapa toteuttaa kaikki yllä mainitut asiat. Toteutustavan huolellisella valinnalla, sekä hyvällä ulkoasun suunnittelulla on mahdollista toteuttaa hyvin toimiva käyttöliittymä prosessille, joka on näinkin monimutkainen.

Seuraavissa kappaleissa käydään läpi opinnäytetyössä kehitettävän käyttöliittymän kannalta oleellisia teknologioita. Käyttöliittymän koodaamiseen käytetään Visual Studio Codea. Käyttöliittymän rakenne, painikkeet ja ulkoasut tehdään HTML- ja CSS kielillä. Jotta prosessin voi suorittaa alusta loppuun käyttöliittymässä, niin siitä on pystyttävä suorittamaan Python -kielellä toimivia koodeja. Näistä koodeista oleellisimmat ovat datan päivittäminen verkkosivujen haravoinnin avulla ja datan käsittely, sekä vieminen simulaatioon ja simulaation lopputuloksen näyttäminen käyttäjälle.

Visual Studio Code

Visual studio code on Microsoftin luoma tekstinkäsittelyeditori, joka on tarkoitettu optimoitu sovellusten luomiseen ja muokkaamiseen. Kyseessä on ilmainen sovellus, jonka voi joko ladata omalle koneelle, tai sitä voi käyttää selaimessa. Sovellus tukee monia eri ohjelmointikieliä, esimerkiksi HTML, CSS ja Python, joita käytetään tässä opinnäytetyössä. Ohjelmassa on myös mahdollista laittaa koodin korjaaminen päälle, mikä on hyödyllistä, jotta virheet tulevat helposti näkyviin, eikä niitä aina joudu etsimään itse koodista. Sovellukseen voi näiden lisäksi ladata todella paljon erilaisia lisäosia tukemaan monia erilaisia tapoja kehittää omaa sovellustaan. Hellerin (2022) mukaan lisäosia on artikkelin kirjoittamishetkellä ollut suunnilleen 38000 erilaista. (Heller 2022; Pedamkar. N.d.)

HTML

Koladen (2021) mukaan HTML (Hypertext Markup Language) on vuonna 1991 julkaistu merkintäkieli, joka määrittelee rakennetta web-sivuilla. Sitä käytetään käytännössä jokaisella nettisivulla näyttämään teksti halutulla tavalla ja halutuissa kohdissa. Pelkästään tekstin rakenteella ei saada nettisivuja näyttämään hyvältä, tai muokattua niitä näytön koon mukaan. HTML antaa tekstille vain rakenteen, mutta ei ulkoasuja.

CSS

CSS (Cascade Style Sheets) on vuonna 1996 julkaistu tyyliohjekieli. Sitä käytetään antamaan HTML-teksteille paremmat ja näyttävämmät ulkoasut. Sen käyttö mahdollistaa paremmin responsiivisten sivujen tuottamisen, kuin pelkän HTML:n käyttäminen. Pääasiassa kieltä käytetään tuottamaan visuaalisesti parempia nettisivustoja, jotka toimivat näyttökoosta riippumatta. (What is CSS. N.d.)

Verkkosivujen haravointi

Verkkosivujen haravointi ("data scraping") tarkoittaa tiedon automaattista hakemista internetsivustolta ihmiselle luettavaan muotoon. Se on esimerkiksi hyödyllistä silloin, kun halutaan hakea jostain isommasta taulukosta vain tietyt rivit ja laittaa ne talteen jonnekin muualle. (Terminfo, 2014)

4.3 Käyttöliittymää varten tutkittavat teknologiat

Pythonia on mahdollista kirjoittaa HTML -tiedostoihin ilman ohjelmistokehystä, mutta yleisesti sitä ei pidetä hyvänä vaihtoehtona, sillä se saattaa tuottaa ongelmia koodien hallinnassa tai ylläpitämisessä. Seuraavissa kappaleissa käydään läpi, mikä on ohjelmistokehys ja kolme vaihtoehtoa Pythoniin liittyvistä viitekehyksistä, joista mikä tahansa voidaan valita käytettäväksi prosessia päivittäessä.

Ohjelmistokehys

Ohjelmistokehys on rakenne sovelluskehitykselle, joka toimii sovelluksen kehittämisen aloituspisteinä. Sen tarkoituksena on säästää aikaa ja vähentää virheitä etukäteen suunnitellulla ja testatulla rakenteella. Käyttämällä ohjelmistokehystä kehittäjiä ei tarvitse alkaa rakentamaan sovellusta täysin puhtaalta pöydältä, vaan koodia voi alkaa kirjoittamaan jo tiedossa oleviin paikkoihin. Viitekehysten käyttäminen myös helpottaa testaamista ja virheiden löytämistä. Tiivistettynä ohjelmistokehys on olennainen työkalu kehittäjille rakentaa monimutkaisia sovelluksia tehokkaammin. (What Is a Framework, 2021)

PyScript

PyScript on ohjelmistokehys, joka mahdollistaa pythonin käyttämisen HTML- tiedostoista helposti ja nopeasti. Se tulee olemaan yksi vaihtoehtoista, joita käydään läpi opinnäytetyön käyttöliittymän toteutustapaa tutkittaessa. Se mahdollistaa Pythonin käytön selaimessa, sekä parantaa Pythonin ja Javascriptin välistä kommunikaatiota. (Ulili 2022.)

Flask

Flask on mikro web- ohjelmistokehys, joka on kirjoitettu Pythonilla. Se on määritelty mikro- koon viite-kehykseksi, koska se ei vaadi erityisiä työkaluja, tai kirjastoja. Siinä ei ole validaatioita, tai komponentteja mistä hakea kolmannen osapuolen luomia kirjastoja normaaleihin funktioihin. Flask on kevyt ohjelmistokehys, jonka kanssa on helppo päästä alkuun. Sitä käytetään usein pieniä tai keskikokoisia web-sovelluksia luodessa. Se tarjoaa helppokäyttöisen ja yksinkertaisen API:n joka mahdollistaa sovellusten luomisen nopeasti ja helposti. Flask mahdollistaa monenlaisien sovellusten luomisen helpoista nettisivuista monipuolisiin ja haastaviin sovelluksiin. (Great Learning Team 2022.)

Django

Django on korkeatasoinen Python- ohjelmistokehys, joka mahdollistaa turvallisten ja ylläpidettävien nettisivujen kehittämisen. Jotkut Djangon osat käyttävät ORM(Object Relational Mapping)-tekniikkaa, automaattista admin käyttöliittymää ja sisäänrakennettua tukea käyttäjän tunnistautumiselle ja valtuuksille. Djangolla on myös suurehko ja aktiivinen yhteisö, joka tuottaa ja tukee kolmannen osapuolen kirjastoja. (Django Software Foundation n.d.)

5 Todennäköisyslaskennan kehittäminen

5.1 Prosessin tilanne alkuvaiheessa

Opinnäytetyön alussa prosessilla ei ole olemassa käyttöliittymää. Dataa haetaan joko käsin, tai pythonilla kirjoitetun verkkosivujen haravointi -koodin avulla. Tämän jälkeen dataa käsitellään monessa eri Excel-- sivussa, jonka jälkeen käsitelty data viedään pythonilla tehtyyn koodiin, joka laskee tietystä ottelusta todennäköisyydet tapahtumille. Koodi käyttää molempien joukkueiden Exceliin laskettuja voimalukuja useista eri kategorioista, jonka jälkeen se simuloi jääkiekko-ottelun tapahtumia. Itse simulaatiossa otetaan huomioon esimerkiksi laukaisutilanteiden määrä ja laatu, pelivälineen hallinta ja ylivoimat. Mikäli simulaatioon antaa molempien joukkueiden kaikiksi arvoiksi 0, menevät arviot kaikilla osa-alueilla lähes tasan molemmille joukkueille ja ne noudattavan NHL:än keskiarvoja kyseisistä kategorioista. Arvot eivät mene täysin tasan molemmille joukkueille, koska simulaatio ottaa huomioon sattuman, joka on satunnainen jokaisessa arvioidussa tapahtumassa ottelun sisällä. Merkityksellisimmät arvot, joita simulaatio antaa, on maaliodottama ja maalien kokonaismäärä.

Kyseessä on siis monimutkainen prosessi, jolle pitäisi saada luotua mahdollisimman yksinkertainen ja helppokäyttöinen käyttöliittymä. Prosessia suorittaessa on oltava mahdollista muokata dataa ennen kuin se päättyy simulaatioon, valita joukkueet, sekä nähdä valittujen joukkueiden lopulliset todennäköisyydet vähintään maaliodottaman ja voittotodennäköisyyksien suhteen, kun laskenta on suoritettu. Käyttöliittymää luotaessa selvitetään mahdolliset vaihtoehdot ja valitaan paras juuri tätä prosessia varten. Alla olevassa kuvassa näkyy simulaation tuottama tuloste ja arviot ottelun tapahtumista maalimäärien, jäähyjen ja ottelun lopputuloksen kannalta. Näistä lukemista oleellisin arvo on tehtyjen maalien odotusarvo ottelua kohden, jonka avulla voidaan laskea todennäköisyys ottelun voittajasta.

```

PHI maalimäärien prosenttijakauma
0.0724 0.1977 0.2582 0.219 0.1411 0.068 0.0288 0.0101 0.0031 0.0013 0.0003

VGK maalimäärien prosenttijakauma
0.0489 0.1522 0.229 0.2249 0.1672 0.0979 0.0486 0.0193 0.0085 0.0027 0.0001

Jäähyminuutit keskimäärin/Ottelu

PHI 4.6711

VGK 4.4763

PHI Tehtyjen maalien odotusarvo 2.5585

VGK Tehtyjen maalien odotusarvo 2.9704

43.03 % Kotivoiton todennäköisyys

56.97 % Vierasvoiton todennäköisyys

```

Kuvio 4. Simulaation tulostama todennäköisyysarvio.

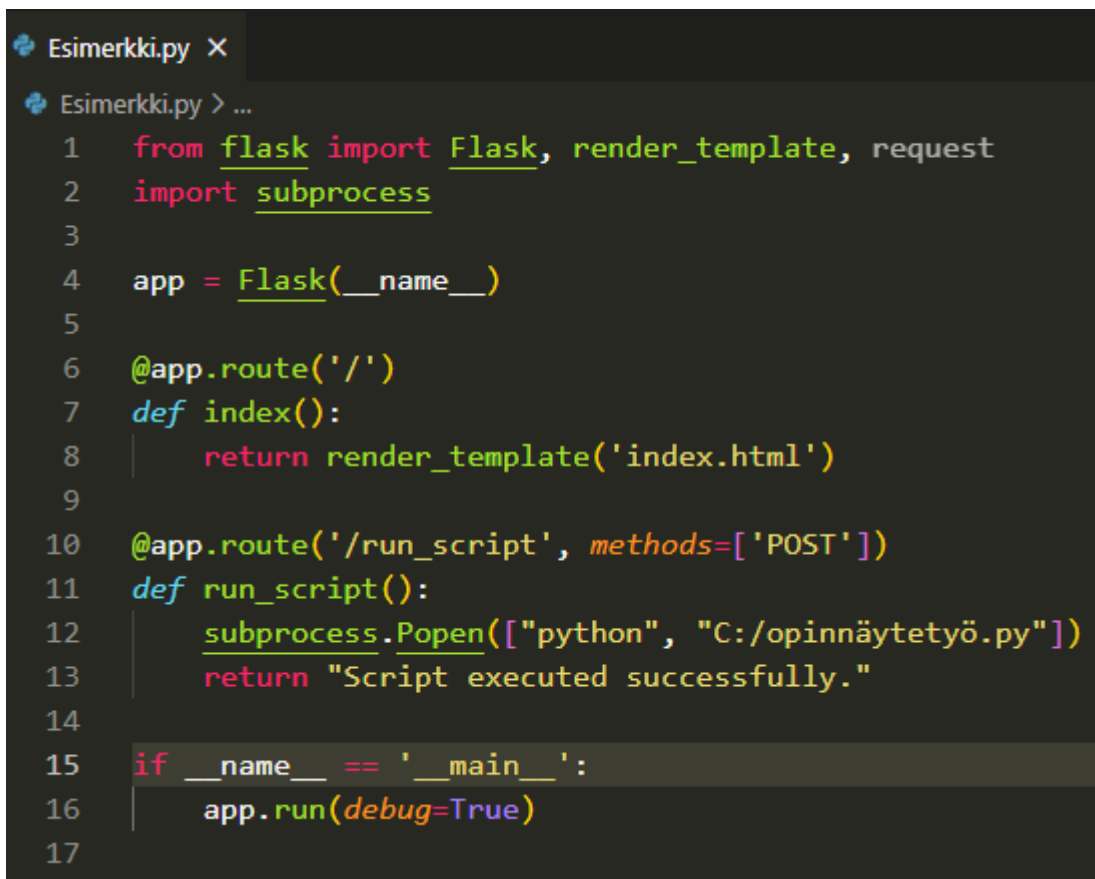
5.2 Prosessin kehittäminen sovellukseksi

Tässä luvussa käydään vaihe vaiheelta läpi, mitä kaikkea käytiin läpi sovelluskehityksen yhteydessä, siihen liittyviä haasteita, sekä perustellaan ratkaisut joihin päädyttiin. Kehittämistyö alkoi prosessin huolellisella analyysillä, sekä kaikkien kaavojen tarkastamisella. Prosessin kannalta oli äärimmäisen tärkeää, että kaikki voimaluvut joita lasketaan tilastoista, tuotetaan täsmälleen samalla tavalla. Mikäli niiden laskutapaa olisi lähdetty muokkaamaan, ei olisi mahdollista toteuttaa millään tavalla realistista vertailua aloituspisteen ja lopputuotoksen välillä. Kehittämistyötä aloittaessa prosessin vieminen alusta loppuun vei aikaa yli 10 minuuttia. Tähän aikaan sisältyy siis yksittäisen ottelun todennäköisyysarvio, joka kerrotaan maksimissaan kuudellatoista, mikäli NHL:ssä pelataan täysi kierros otteluita. Laskennallinen aika voi siis päivittäin olla maksimissaan 2,5-3 tuntia.

Kun oli tiedossa mitä tarkalleen pitää pystyä kehittämään, niin alkoi työvaiheiden ja lopputuotoksen suunnittelu. Ensimmäiseksi suunniteltiin, mitä nappeja käyttäjä tarvitsee. Aluksi oli pystyttävä valitsemaan joukkueet ja varmistamaan valinta. Sen jälkeen käyttäjän on nähtävä, mitä dataa jouk-

kueista on kerätty ja laskettu voimaluvuiksi. Tässä vaiheessa käyttäjän on vielä pystyttävä muokkaamaan voimalukuja joko käyttöliittymästä, tai mahdollisesti käsin Excelistä. Kun voimaluvut ovat kunnossa, on käyttäjän pystyttävä painamaan nappia, josta voimaluvut viedään simulaatioon, joka näyttää käyttäjälle vähintään oleelliset arviot. Tämän jälkeen prosessi on yhden ottelun osalta suoritettu, joten olisi hyvä päästä myös napin painalluksella prosessin alkupisteeseen. Oleellista opinnäytetyön kannalta oli myös se, että joukkueiden valinnan jälkeen tapahtuu päivitetyn datan hakeminen suoraan datalähteestä, eikä esimerkiksi edellisen päivän Excel -tiedostosta.

Kun käyttöliittymän suunnittelu oli siinä pisteessä, että tiedettiin tarvittavat vaatimukset toimivalle ratkaisulle, oli seuraavana työvaiheena kartoittaa, mitä mahdollisuuksia on olemassa datan käsittelylle käyttöliittymästä Python -kielen kanssa. Perusteellisten tutkimusten jälkeen selkeästi paras vaihtoehto oli toteuttaa prosessi viitekehyksen avulla, koska sovellusta ei tarvitse lähteä rakentamaan nollasta ja oppaita aiheeseen löytyy lukemattomat määrät internetistä. Viitekehykseksi valikoitui Flask. Lisää viitekehyksestä sekä viitekehyksen valinnasta löytyy seuraavassa luvussa. Flask oli minulle uusi teknologia, mutta hyvin nopeasti kävi selväksi, että kyseinen ohjelmistokehys oli äärimmäisen helppo ottaa käyttöön, sekä suhteellisen yksinkertainen. Alla kuva, jossa on Flask:in avulla toteutettu sovellus, jonka saa ajettua käyttöliittymästä. Muutaman kuvassa olevan koodirivin lisäksi tarvitsi vain luoda Python -tiedosto jota ajaa, sekä HTML -tiedosto, joka tulkkaa sovelluksen käyttöliittymäksi.



```

Esimerkki.py X
Esimerkki.py > ...
1  from flask import Flask, render_template, request
2  import subprocess
3
4  app = Flask(__name__)
5
6  @app.route('/')
7  def index():
8      return render_template('index.html')
9
10 @app.route('/run_script', methods=['POST'])
11 def run_script():
12     subprocess.Popen(["python", "C:/opinnäytetyö.py"])
13     return "Script executed successfully."
14
15 if __name__ == '__main__':
16     app.run(debug=True)
17

```

Kuvio 5. Esimerkkikuva Flaskin käytöstä.

Kun ohjelmistokehitys oli saatu testattua ja toteutettua, oli aika siirtyä selvittämään, mikä olisi paras tapa hakea datat internetistä niin, että ne olisivat mahdollista saada siirrettyä suoraan simulaatioon. Prosessin alkutilanteessa tämä osio oli suoritettu hakemalla datat lähteestä haravoimalla verkkosivuja ja siirtämällä kaikki data jokaisesta NHL -pelaajasta samaan tiedostoon. Tämän jälkeen pelaajat jaettiin käsin omiin joukkueisiinsa eri Excelin välilehtiin. Seuraavat vaiheet ennen voimalukujen simulaatioon viemistä oli ensin laskea ne käsin jokaiselle joukkueelle, mikä vei erittäin paljon aikaa. Tämän jälkeen jokaisen joukkueen voimaluvut vielä vietiin samaan Excel -tiedostoon, josta simulaatio sai haettua datat ja laskettua todennäköisyydet.

Datan käsittelyn osalta tutkittiin mahdolliset vaihtoehdot ja parhaaksi vaihtoehdoksi osoittautui Pandas -kirjasto. Kyseistä kirjastoa on käsitelty tässä opinnäytetyössä luvussa 4.2. Hyvien datankäsittelyvaihtoehtojen lisäksi, minulta löytyi aiempaa kokemusta kirjaston käytöstä, joten kyseessä oli suhteellisen helppo valinta. Pandas:in käyttöönotto on myös tehty todella helpoksi, kuten alla

olevasta esimerkistä näkyy. Kuvassa on tuotu Pandas osaksi Python -tiedostoa ja sille on määritetty sivusto, josta halutaan hakea dataa talteen.

```

1  import pandas as pd
2  import openpyxl
3  from openpyxl.utils.dataframe import dataframe_to_rows
4
5
6  url = 'https://www.naturalstattribick.com/playerteams.php?'
7  df = pd.read_html(url)[0]
8  |
9  wb = openpyxl.Workbook()
10
11  ws = wb.create_sheet("My Worksheet")
12
13
14  team_names = df["Team"].unique()
15  results = {}
16  for team_name in team_names:
17      df_filtered = df.loc[df["Team"] == team_name]

```

Kuvio 6. Esimerkkikuva Pandas -kirjaston käytöstä.

Kun datan hakeminen oli testattu, seuraava vaihe oli ratkaista se, millä tavalla haetaan juuri se oikea data mitä käyttäjä haluaa. Koska kahden joukkueen pelaajatilastoissa on suunnilleen 40-50 pelaajaa ja koko NHL:n pelaajatilastoissa on 950+ pelaajaa, ei ole mitään järkeä käyttää aikaa siihen, että haetaan paljon pelaajia, joiden dataa ei käytetä yksittäisen ottelun arviossa. Vaihtoehtoina oli joko antaa käyttäjän kirjoittaa halutut joukkueet, tehdä jokaiselle joukkueelle omat napit, tai luoda pudotusvalikko, josta käyttäjä voi valita haluamansa joukkueet. Näistä kolmesta vaihtoehdosta viimeinen oli selkeästi paras, sillä esimerkiksi hakutilanteessa kirjoitusvirheet ovat mahdollisia ja toisaalta jokaiselle joukkueelle oman napin (32 nappia) luominen näytölle ei ole käyttäjäystävällinen vaihtoehto. Pudotusvalikko vaati paljon manuaalista kirjoittamista, mutta se on ainoa selkeästi hyvä vaihtoehto olemassa olevista. Alla kuva koodista, jolla se luotiin.

```

<body>
  <h1>Valitse joukkueet:</h1>
  <form method="POST" action="/">
    <label for="team1" class="isompi">Koti:</label>
    <select name="team1" id="team1">
      <option value="BOS">Boston Bruins</option>
      <option value="MTL">Montreal Canadiens</option>
      <option value="COL">Colorado Avalance</option>
      <option value="CHI">Chicago Blackhawks</option>
      <option value="PHI">Philadelphia Flyers</option>
      <option value="CAR">Carolina Hurricanes</option>
      <option value="NYR">New York Rangers</option>
      <option value="N.J">New Jersey Devils</option>
      <option value="T.B">Tampa Bay Lightning</option>
      <option value="TOR">Toronto Maple Leafs</option>
      <option value="VGK">Vegas Golden Knights</option>
      <option value="WPG">Winnipeg Jets</option>
    </select>
  </form>

```

Kuvio 7. Esimerkkikuva HTML -tiedostosta.

Seuraava vaihe oli muokata aiemmin Excelissä suoritettut laskukaavat toimimaan Pythonin avulla. Tämä vaihe ei juurikaan tuottanut sen isompia vaikeuksia, koska Excelissä käytetyissä kaavoissa oli kyse laskuista, jotka onnistuvat perustoiminnoilla. Pythonissa ne oli helppo toteuttaa, kun oli selvillä, että mitä tarkalleen pitää tehdä.

Kun datanhaku ja käsittely voimalukuihin asti oli suoritettu, niin täytyi alkaa suunnittelemaan käyttöliittymän käytännön toteutusta. Sitä, miten data tulee esittää ja miten sitä voi muokata voimalukujen näyttämisen jälkeen. Koska pääosin en koe tarpeelliseksi muokata dataa siinä vaiheessa, kun laskennat on suoritettu, en kokenut tarpeelliseksi mahdollistaa sen muokkaamista käyttöliittymässä. Sen sijaan laitoin voimaluvut omaan Exceliin, johon ei tule mitään muuta kuin otsikot ja simulaatiossa tarvittavat luvut. Käyttöliittymä hakee voimaluvut käyttäjälle näkyviin heti, kun ne ilmestyvät Exceliin ja näyttää ne HTML -taulukkona. Mikäli lukuja on tarve vaihtaa, käyttäjä voi avata Excel -tiedoston ja vaihtaa manuaalisesti haluamaansa lukua. Kyseiseen toimenpiteeseen menee alle 10 sekuntia.

Kun käyttäjä on tarkastanut voimaluvut, tarvitsee enää painaa yhtä nappulaa, joka vie voimaluvut simulaatioon. Tämä vaihe tuotti hieman ongelmia, sillä simulaation läpi ajaminen kestää muutamia sekunteja, joten ensimmäisillä testikerroilla simulaation lopputulokset näyttävä sivu ehti hakea edellisen ottelun todennäköisyysarviot ja näyttää ne sen sijaan, että valittujen joukkueiden keskinäisestä ottelusta olisi näytetty mitään dataa. Ongelma oli odotettua monimutkaisempi, mutta se ratkesi lopulta todella yksinkertaisella koodilla, joka kääntää sovellusta odottamaan, kunnes halutussa tiedostossa tapahtuu muutos. Alla kuva koodista, joka ratkaisi ongelman.

```
file_path = 'statistics.xlsx'
last_modified = os.path.getmtime(file_path)

while True:
    current_modified = os.path.getmtime(file_path)
    if current_modified > last_modified:
        df6 = pd.read_excel(file_path)
        break
    else:
        time.sleep(1)
```

Kuvio 8. Esimerkkikuva Python -tiedostosta.

Viimeinen vaihe prosessin päivittämisestä oli käyttöliittymän viimeistely ja sen muokkaaminen mahdollisimman helppokäyttöiseksi sekä selkeäksi. Tämä vaihe oli helpohko ja nopea. Mielestäni käyttöliittymästä tuli erittäin selkeä, eikä siinä ole nappeja mistä voi tehdä peruuttamattomia virheitä. Alla kuvat käyttöliittymän näytöistä.

Select Teams

localhost:5000

Valitse joukkueet:

Koti: Winnipeg Jets

Vieras: Winnipeg Jets

Boston Bruins
 Montreal Canadiens
 Colorado Avalance
 Chicago Blackhawks
 Philadelphia Flyers
 Carolina Hurricanes
 New York Rangers
 New Jersey Devils
 Tampa Bay Lightning
 Toronto Maple Leafs

Valitse

Kuvio 9. Kuva käyttöliittymästä 1.

Ensimmäisellä näytöllä on kaksi pudotusvalikkoa, joista valitaan koti- ja vierasjoukkue. Kun joukkueet on valittu, painetaan "Valitse" -nappia, mikä käynnistää datan hakemisen ja käsittelyn. Kun data on käsitelty, aukeaa seuraava näyttö.

Voimaluvut

NYR - N.J

Kategoria	Koti	Vieras
Nimi	NYR	N.J
MV	1001	996
XG	44	56
Laukominen	1001	997
AV	1005	1004
Tempo	435	435
Ylivoima	1005	1008

Simulaatio

Kuvio 10. Kuva käyttöliittymästä 2.

Toisella näytöllä näkyy voimaluvut ja valitut joukkueet. Tilastot ovat tehty mahdollisimman selkeiksi luettaviksi ja asetettu keskelle näyttöä. Tässä vaiheessa on vielä mahdollista muuttaa voimalukuja käsin Excelistä, mikäli niin haluaa tehdä.

Simulaation tulokset

Joukkue	NYR	N.J
Tehtyjen maalien odotusarvo	3.0147	2.9663
Voiton todennäköisyys	51.1000	48.9000

[Etusivulle](#)

Kuvio 11. Kuva käyttöliittymästä 3.

Sovelluksen viimeinen näyttö. Kuvassa on simulaation tulokset, joista näkyy maaliodottama, sekä ottelun todennäköisyydet. Yksittäisen ottelun todennäköisyyksien arvioimiseen menee uudistetun prosessin avulla datan päivittämisen kanssa huomattavasti vähemmän aikaa.

6 Tutkimustulokset ja pohdinta

Tässä luvussa käydään läpi oleelliset asiat tuloksista sekä pohditaan opinnäytetyön aikana toteutettuja tutkimuksia. Luvussa kerrotaan, kuinka tutkimusongelmat ratkaistiin ja minkä takia ne on ratkaistu kyseisillä menetelmillä. Lopuksi käydään läpi opinnäytetyön merkitystä ja tuloksia.

6.1 Käyttöliittymän toteutus

Ensimmäinen tutkimusongelma oli seuraavanlainen; Kuinka toteuttaa käyttöliittymä, josta käyttäjä voi arvioida tiettyjen joukkueiden voimasuhteita? Ongelma liittyi pääasiassa ohjelmistokehityksen löytämiseen, joka mahdollistaisi Pythonin käyttämisen käyttöliittymästä opinnäytetyössä parannetun menetelmän vaatimuksien mukaan.

Opinnäytetyössä käsiteltävää prosessia varten tutkittiin kolmea erilaista Pythoniin liittyvää ohjelmistokehitystä. Tutkimuksen kohteina olivat Django, Flask ja PyScript. Django ja Flask ovat molemmat olleet olemassa jo pidemmän aikaa Python-kehityksessä käyttöliittymien kanssa ja PyScript on suhteellisen uutta teknologiaa niihin verrattuna. Näiden kolmen ratkaisun väliltä etsittiin juuri tämän prosessin kannalta parasta teknologiaa, joten valinnassa painotettiin vahvasti helppokäyttöisyyttä, suorituskykyä ja muokattavuutta.

Opinnäytetyössä käyttöön otettavaksi viitekehikseksi valikoitui Flask. Näistä kolmesta vaihtoehdosta se vastaa parhaiten parhaiten opinnäytetyössä käsiteltävän prosessin vaatimuksia. Flask on Djangoon ja PyScriptiin verrattuna kevyempi ratkaisu, jota on helppo käyttää. Kun sitä vertaa Djangoon, se on paljon joustavampi ja helpommin muokattavissa, mikä tekee käyttöliittymän kehittämisestä tarpeita vastaavaksi helpompaa. Flask on myös parempi ratkaisu suorituskyvyn kannalta. Django ja PyScript ovat myös tehokkaita, mutta niiden suorituskyky ei ole yhtä hyvä kuin Flaskilla niiden suuremman koon ja monimutkaisen arkkitehtuurin vuoksi. PyScript on myös suhteellisen helppo ottaa käyttöön, mutta Flaskiin tai Djangoon verrattuna sillä on paljon pienempi yhteisö kehittäjiä, joka vaikuttaa käytettävissä olevien kirjastojen määrään merkittävästi.

6.2 Datan käsittely Pythonin avulla

Toisessa tutkimusongelmassa etsittiin vastausta siihen, kuinka automatisoida datankäsittely mahdollisimman pitkälle. Alkuperäisessä menetelmässä data käsiteltiin lähes kokonaan Excelissä, jonka jälkeen valmiiksi käsitellyt numerot vietiin simulaatioon todennäköisyyksien laskemista varten.

Sovelluksen nopeuttamisen kannalta oli oleellista minimoida Excelissä tapahtuvien tehtävien määrää, joten Pandas oli luonnollinen vaihtoehto käsitellä dataa uudessa sovelluksessa mahdollisimman paljon. Kyseinen kirjasto tarjoaa jopa Exceliä enemmän vaihtoehtoja datan käsittelyyn ja se myös nopeuttaa datan viemistä simulaatioon asti. Alkuperäisessä menetelmässä kyseistä kirjastoa oli käytetty hyvin vähän verrattuna siihen, miten paljon sillä oli mahdollista lopulta nopeuttaa prosessia. Pandasin avulla datataulukon pystyi hakemaan suoraan verkosta ja asettamaan vaadittavat numerot muuttujiin, joita oli helppo laskea keskenään pythonin normaaleilla matemaattisilla toiminnoilla.

Datan käsittelyn suhteen tutkimustyötä jouduttiin tekemän käyttöliittymään verrattuna huomattavasti vähemmän. Pandas -kirjastoa tutkimalla selvisi hyvin nopeasti, että kaikki vaadittavat toimenpiteet olisi mahdollista toteuttaa sen avulla. Lisäksi kirjasto oli tuttu ennestään, koska sitä oli käytetty jonkin verran alkuperäisessä simulaatiossa.

6.3 Kehittämistyön tulokset

Opinnäytetyön aikana kehitettiin tavoitteisiin päässyt toimiva käyttöliittymä, josta käyttäjä voi valita haluamansa NHL- joukkueet todennäköisyysarvion tuottamiseen. Käyttöliittymästä tuli tavoitteiden mukaisesti erittäin helppo käyttää ja manuaalista todennäköisyysarviota huomattavasti nopeampi. Yksittäisen ottelun arvioimiseen sovellusta käyttämällä aikaa kuluu keskimäärin noin 20 sekuntia. Tämä on merkittävästi nopeampi aika, jos verrataan alkuperäiseen menetelmään, joka vei aikaa yksittäistä ottelua kohden hieman päälle kymmenen minuuttia. Kun lasketaan ajan säästämistä isommassa mittakaavassa, niin voidaan käyttää esimerkkinä NHL:n runkosarjaa. Sitä pelataan yhteensä 1312 yksittäistä ottelua. Jos lasketaan, että ottelua kohden aikaa säästyy kymmenen minuuttia, tulee yhden kauden aikana säästettyä lähes 220 tuntia.

Ajan säästäminen ei ole ainoa asia, jota sovellus parantaa vanhaan menetelmään nähden. Sovelluksessa dataa käsitellään automaattisesti, joten käyttäjän tekemiä virheitä ei tule. Virheiden poistamisen lisäksi sovellus on niin helppo käyttää, että uudelle käyttäjälle ei tuota ongelmia ymmärtää miten sovellus toimii.

6.4 Opinnäytetyön eettisyys ja luotettavuus

Opinnäytetyö on tehty noudattaen hyviä tieteellisiä käytäntöjä. Opinnäytetyössä ei ole kerätty arkaluontoista materiaalia tai henkilötietoja, sillä useille opinnäytetöille tyypillisiä kyselytutkimuksia ei tarvittu opinnäytetyön materiaalia kerätessä, joten opinnäytetyö on eettisesti tuotettu. Opinnäytetyössä kehitetty sovellus antaa samat numerot mitä manuaalisesti tehtävä arvio, joten kehittämistyön tulokset on saatu todennetusti tehtyä luotettavasti ja tavoitteiden mukaan. Opinnäytetyössä käytetyissä lähteissä on priorisoitu suoraan kehittäjiltä tulleita materiaaleja, joten lähteet ovat myös luotettavia.

6.5 Jatkotoimenpiteet

Opinnäytetyön aikana tehty sovellus toimii hyvänä pohjana mahdolliselle jatkokehittämiselle. Mikäli sovellusta haluaa kaupallistaa, on siihen mahdollista lisätä kirjautuminen ja lisää eri lajeja. Mahdollinen kirjautumisen ja lajivalikoiman lisääminen antaisi eri käyttäjille mahdollisuudet valita haluamansa lajit sovelluksesta ja tuottaa arvioita ainoastaan niistä lajeista, joista haluavat. Sovelluksen jatkokehittäminen ei kuitenkaan ole välttämätöntä, sillä se antaa jo todennäköisyydet NHL-otteluista.

Lähteet

Codecademy. 2021. What is a Framework. Viitattu 24.4.2023. <https://www.codecademy.com/resources/blog/what-is-a-framework/>

Django Software Foundation. N.d. Viitattu 24.4.2023 <https://www.djangoproject.com/>

Geldart, M. 2022. WebsiteBuilderInsider. How do I plan a UI UX project?. Viitattu 26.4.2023 <https://www.websitebuilderinsider.com/how-do-i-plan-a-ui-ux-project/>

Great Learning Team. 2022. A Brief Introduction to Pandas – What is Pandas in Python. Viitattu 24.2.2023. <https://www.mygreatlearning.com/blog/python-pandas-tutorial/>

Great Learning Team. 2022. Everything you need to know about Flask for beginners. Viitattu 12.4.2023 <https://www.mygreatlearning.com/blog/everything-you-need-to-know-about-flask-for-beginners/>

Gröhn, J., Huusko, J-M., Juvonen, M. ja Pirhonen, E. N.d. Mahtavaa matematiikkaa. Viitattu 19.11.2022 <https://sites.uef.fi/mahtavaa-matematiikkaa/todennakoisyys/>

Heller, M. 2022. InfoWorld. What is Visual Studio Code? Microsoft's extensible code editor. Viitattu 13.1.2023. <https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html>

Javatpoint. N.d. What is CSS. Viitattu 12.2.2023 <https://www.javatpoint.com/what-is-css>

Kananen, J. 2015. Kehittämistutkimuksen kirjoittamisen käytännön opas: miten kirjoitan kehittämistutkimuksen vaihe vaiheelta. Tampere: Suomen yliopistopaino – Juvenes Print.

Kolade, C. 2021. Freecodecamp. What is HTML – Definition and Meaning of Hypertext Markup Language. Viitattu 3.2.2023 <https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/>

Libraries in Python. 2021. Geeksforgeeks. Viitattu 24.2.2023. <https://www.geeksforgeeks.org/libraries-in-python/>

Nordicbet. 2020. Mikä on xG eli maaliodottama ja miten se vaikuttaa vedonlyöntiin. Viitattu 8.1.2023. <https://www.nordicbet.com/fi/blogi/pitkavetovihjeet/vedonlyontiopas/xg-eli-maaliodottama/>

Pedamkar, P. N.d. Educba. What is Visual Studio Code?. Viitattu 9.1.2023. <https://www.educba.com/what-is-visual-studio-code/>

Piccolo, N. 2022. Inside the stats. Viitattu 1.5.2023. <https://insidetherink.com/inside-the-stats-expected-goals/>

Python vs Excel for Data Analysis. 2022. Cambridge Spark. Viitattu 23.2.2023. <https://www.cambridgespark.com/info/python-vs-excel>

Sanastokeskus. 2014. Terminfo -verkkolehti 1/2014. Termiharava. Viitattu 8.1.2023. <http://www.terminfo.fi/sisalto/termiharava-42.html>

Selvaraj, N. 2023. Python Excel Tutorial: The Definitive Guide. Datacamp. Viitattu 23.2.2023. <https://www.datacamp.com/tutorial/python-excel-tutorial>

Stanley Ulili. 2022. Intro to PyScript: Run Python in the browser. Viitattu 5.2.2023 <https://blog.logrocket.com/pyscript-run-python-browser/>

W3schools. N.d. Python Introduction. Viitattu 5.2.2023 https://www.w3schools.com/python/python_intro.asp

What is NumPy. N.d. NumPy. Viitattu 24.2.2023. <https://numpy.org/doc/stable/user/whatis-numpy.html>

Yle. 2013. Todennäköisyyslaskenta ja tilastot. Viitattu 1.5.2023 <https://yle.fi/aihe/artikkeli/2013/06/12/todennakoisyysslaskenta-ja-tilastot>

Ylikerroin. 2017. Vedonlyöntisanastoa. Viitattu 8.1.2023. <https://www.ylikerroin.com/oppaat/vedonly%C3%B6nti/vedonly%C3%B6ntisanastoa>