eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Gradu Amaierako Lana

Informatika Ingeniaritzako Gradua

Konputazioa

MLOps teknikaren eta ikasketa inkrementalaren ikerketa eta inplementazioa

Iker Sancho Peinado

Zuzendariak
Aginako Bengoa, Naiara
López González, Roberto

2023.eko irailaren 17

Laburpena

Azken urteetan, gizakiok sortutako datu kopuruak era etengabean hazi egin dira, izan ere, teknologiaren gorakadak eta edozeinek gailu teknologiko desberdinak izateko aukerak egoera hau sendotzen baitute. Honen ondorioz, datu hauetara moldatzen diren modelo prediktiboak sortzea ataza geroz eta zailago da eta ikaskuntza automatikoaren arloan hainbat irtenbide proposatu eta garatu egin dira. Pentsatu behar da kasu gehienetan datu guztien artean garrantzitsuena azkenak direla, izan ere, hauek dira momentuko tendentzia definitzen dutenak. Lan honen bidez, arazo honen aurrean irtenbide partikular bat aurkeztuko da, zehazki, Madrilgo kontaminazioa aurreikusten duen eta *ikasketa inkrementala* deituriko teknikari esker datu berrietara moldatzen den modelo bat sortuko da. Horretarako bi teknika desberdin erakutsi dira datu berriak kontuan hartzea duen garrantzia aztertuz eta bien arteko emaitzak konparatuz.

Bestalde, kontuan izan behar da ere sortutako datuak hain ugariak direnez eta, era berean, hain aldakorrak direnez, eredu hauek denboran zehar gainbegiratuak izatea ezinbestekoa suerta daitekeela kasu gehienetan. Arazo honetarako *MLOps* paradigma aurki dezakegu zeina *DevOps* paradigmatik eratorriz, Adimen Artifizialaren arloan sortzen diren produktu edo irtenbideak nola tratatu behar diren definitzen duen. Paradigma honen barruan sortutako produktuak ziklikoki fase desberdinetatik pasako dira, datuen tratamendutik hasiz eta modeloaren *deployment* fasera iritsi arte honen entrenamendua bitarte izanik, eta fase guzti hauetan produktua gainbegiratu egingo da sortutako errore berrien bila hauek hurrengo faseetara pasa ez daitezen. Prozesu hau ezinbesteko izango da lanaren garapenean Madrilgo kontaminazioa aurreikusten duen modeloa etengabe entrenatu ahal izateko eta bidean sortzen diren akatsak ekiditeko.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

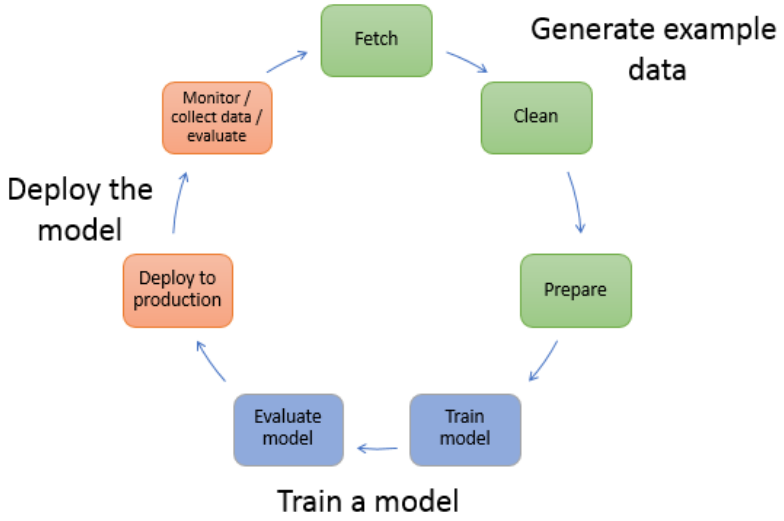
KAPITULUA 1

Sarrera

Azken urteetan Adimen Artifizialaren alorrean izandako gorakadaren ondorioz, adimen artifiziala eta ikaskuntza automatikoa funtsezko tresnak bihurtu dira datu-bolumen handiak prozesatzeko eta hainbat eremutan erabakiak hartzeko. Hala ere, etengabeko bilakaeran dauden datuen prozesamenduak, datuen aldakortasunak eta sistemen baldintzen aldaketek ikaskuntza automatikoko ereduak eguneratzea eska dezakete, haien eraginkortasuna eta zehaztasuna bermatzeko. Testuinguru horretan, **ikaskuntza inkrementala** teknika garrantzitsua bihurtu da ikaskuntza automatikoko ereduak eguneratzeko, eredu osoa hutsetik berriro entrenatzeko beharra deuseztatuz. *Incremental learning*-a 1986tik ezaguna den teknika da **Schlimmer J. C.** eta **Fisher D.**-k garatutako artikuluan ikusi daiteken bezala [4], hala ere, aurretik esan denez gaur egungo testuinguruak erakusten duen datuen momentu oroko aldaketak dira teknika hau azken urteetan hainbesteko garrantzia hartu izana.

Bestalde, garrantzizkoa da aipatzea eredu hauen eraldaketa denboran zehar ikasketa inkrementala erabiliz automatikoki burutzen den prozesua dela kasu gehienetan. Esaterako, datu berrietara moldatzen den eredu batek eguneroko frekuentzia baldin badu (Artikulu honetan erabiliko den froga kasuaren antzera), n datu berri jasotzeko n egun behar izango ditu. Honek ordea beste arazo batera eramaten gaitu: Modeloaren etengabeko eraldapenaren gainbegirapena. Kasu askotan suerta daiteke sarrera datu berriek aurretik ikasitakoarekin talka egitea, eta ondorioz modeloaren predikzioa okerragoa izatea. Horretarako ezinbestekoa izango da ikasketa prozesu osoaren kontrola izatea gainbegirapena zorrotz bat eginik.

Software produktuen garapenaren arloan, proiektuen garapenaren bizi zikloa azkartzeko eta momenturo burutzen ari den fasea gainbegiratzeko ***DevOps*** (*Development Operations*) teknikak hartu du garrantzia, izan ere teknika honen bidez garapen zikliko bat proposatu egiten da non pausu zehatz batzuk era ordenatuan jarraitzen diren bata bestearen atzean prozesuaren etapa guztien gainean kontrola mantentzeko. Noski, teknika honek Machine Learning arlorako bere eratorri propioa du, ***MLOps*** Izenez ezaguna (*Machine Learning Operations*) eta ikasketa inkrementala aplikatzen den prozesuetan berebiziko garrantzia du exekuzio fluxu automatikoa momenturo gainbegiratu ahal izateko. Teknika honek egiten dituen proposamenen artean, datu berrien aurreprozesaketa edota modeloaren monitorizazio etengabea aurki ditzakegu. MLOps kontzeptuaren laburpen bisual bezala 1.1 irudia ikus dezakegu.



1.1 Irudia: MLOps teknikaren eskema kontzeptuala.

Hau guztia esanik eta MLOps teknikan biltzen dituen atalak kontuan izanik, lan honetan bi atal landuko dira bereziki, *deployment* eta *model training*(ziklikoa) atalak zehazki:

- ***Deployment***: Atal hau lantzeko lan modulu bat garatu egingo da (5 atalean azaltzen dena). Honek, modelo baten entrenamendua kudeatzeaz gain, emaitzen entengabeko *deployment* bat mantenduko du entrenatzen joan ahala erantzunen hobekuntza ikus dadin. Gainera, erroreen hurbileko jarraipen bat egingo du.

- ***Model Training***: Bestalde, lan moduluan bertan, modeloaren entrenamendua burutuko da, eta kasu honetan, Ikasketa inkrementala erabiliz entrenatuko dira modeloak. Era honetan, *Model training* ziklikoaren hobekuntzak aski nabariagoak izan dira.

1.1 Helburuak

Lana bere testuinguruan kokaturik dagoela garrantzizkoa da ere hau hau egitearen zergaitia ulertzea eta lortu nahi diren helburuak zeintzuk diren azaltzea. Laburki, lan honetarako pentsatu den helburua *Ikasketa inkrementalaren inplementazio bat lortzea MLOps teknikaz baliatuz prozesuaren exekuzio fluxua momentuoro gainbegiratu ahal izateko* da. Horretarako, bi azpihelburu bereiziko ditugu:

- **Lan modulua**: Alde batetik, ikasketa inkrementala erabiltzen duen modeloa entrenatu ahal izateko, lan modulu zehatz bat sortuko da *Neural Designer* softwarea oinarri bezala erabiliz.

- **Eredu inkrementala**: Bestalde, behin modulua prest dagoela, Madrilgo kontaminazioa aurreikusten duen eta ikasketa inkrementela erabiliz entrenatzen den modeloa sortuko da. Horretarako hainbat balidazio froga burutuko dira.

Inplementazio finala burutu baino lehen, garapen prozesuan zehar hainbat froga burutuak izango dira, bai funtzionamenduaren arlo zehatzak frogatzeko, hala nola emaitzen nondik norakoa aztertzeko. Hau hurrengo atalean azalduko da.

KAPITULUA 2

Proiektuaren kudeaketa

Kapitulu honetan proiektuaren kudeapenarekin zerikusia duten atal guztiak landuko dira, honen deskribapenetik hasita, lanaren plangintza landuz, eta bukatzeko bai arriskuak bai lanean izandako desbiderapenak aztertuz.

2.1 Proiektuaren deskribapena

Gradu Amaierako lanaren proposamenean adierazita zegoenez, proiektuaren deskribapena hurrengoa da:

"Machine Learning Operations, MLOps gisa laburtua eta DevOps (Development Operations) kulturatik eratorrita, joera berrietako bat da Adimen artifizialeko Big Data arloan, eta horregatik, lotura estua du ikaskuntza automatikoko ereduekin (machine learning) eta prozesuen automatizazioan. MLOps-aren helburua, zehazki, enpresa batek duen prozesuari (Machine Learning-aren edo ikaskuntza automatizauaren arloan) eragiketa eredu berri bat aplikatuz, arintzea eta gainbegiratzea da.

Gradu Amaierako Lan proposamen honetan sistema zehatzak garatuko dira adimen artifizialaren arloan (datuen analisirako eta ikaskuntza automatikorako) neural Designer plataformaren barruan."

2.2 Plangintza

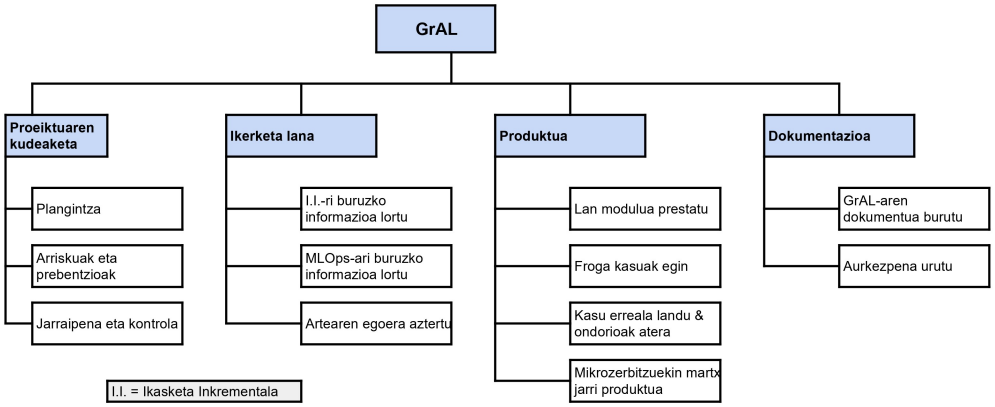
Atal honetan proiektuaren plangintza jasoko da. Plangintza egiteko, proiektuen kudeaketan ezinbestekoak diren atalak landuko dira [5].

LDE diagrama, **Lan paketeak**, **Ordu banaketa**, **Emangarriak**, **Gantt diagrama**, **Desbiderapenak**, **Lan metodologia**.

Atal hauei esker, proiektuaren bizi zikloaren fase desberdinak ondo neurtu, egin beharreko atazen zerrenda garatu eta suerta daitezken arazoak kudeatu (edo ekidin) egingo dira besteak beste.

2.2.1 LDE diagrama

Azpiatal honetan, proiektuaren helburua lortzeko egin behar izan diren lanak edo landu behar izan diren arlo guztien deskonposaketa aztertuko da dagokion LDE diagrama eraikiz 2.1 irudian ikusten den bezala.



2.1 Irudia: LDE: Lan Deskonposaketa Eredua.

Azaldu den bezala, landuko diren blokea nagusiak Proiektuaren kudeaketa bera, aurretiko ikerketa lana, garatu nahi den produktua eta dokumentu hau izango dira. Dokumentu honek, 4 atalen bildura jasoko du eta pausoz pauso nola burutu diren azalduko du ere.

2.2.2 Lan paketeak

Hurrengo pauso honetan, Lan deskonposaketa ereduaren diagraman aztertutako lan pakete guztiak deskribatuko dira banan-banan:

- **Proiektuaren Kudeaketa**:
  - Plangintza: Atal honetan proiektua hasi baino lehen egin beharreko aurrelan guztiak deskribatu eta landuko dira.
  - Arriskuak eta prebentzioak: Hemen, proeiktuak ekarri ditzaken arrisku nagusienak ikertuko dira eta irtenbide egokienak proposatuko dira.
  - Jarraipena eta kontrola: Ataza honi esker, plangintzan aurreikusitako lan banaketak, ordu banaketak, ... konparatu egingo dira proeiktua bukatu ondoren jasotako datuekin konparatuz.
- **Ikerketa Lana**:
  - Ikasketa Inkrementalari buruzko informazio bilketa: Ikasketa Inkrementalaren kontzeptua landuko da eta teknika honek nola funtzionatzen duen azalduko da.
  - MLOps-ari buruzko informazio bilketa: MLOps kontzeptua landuko da eta teknika honek nola funtzionatzen duen azalduko da.
  - Artearen egoeraren azterketa: Ikasketa Inkrementala eta MLOps teknikak gaur egun erabiltzen diren ala ez, eta horrela bada, nola erabiltzen diren ikertuko da.

- **Produktua**:
  - Lan modulua prestatu: Ikasketa inkrementala aurrera eramateko beharrezkoa den lan moduluaren prestaketa azalduko da.
  - Froga kasuak egin: Lan moduluak bere funtzioak ondo burutzen dituelaz zihurtatzeko egindako froga kasuak azaldu eta landuko dira.
  - Kasu erreala landu eta ondorioak atera: Ikasketa inkrementalaren hala nola MLOps tekniken eraginak gaur egun garrantzia duen kasu batean aplikatuz lortutako emaitzak eta ondorioak aztertuko dira.
  - Mikrozerbitzuekin martxan jarri: MLOps teknika guztiz finkatzeko, Lan modulua mikrozerbitzuekin integratzeko aukera erakutsi da hau docker kontainer batean sartuz.
- **Dokumetazioa**:
  - GrAL-aren dokumentua burutu: Egindako lan guztia pausoz pausu dokumentatu egingo da atal honetan.
  - Aurkezpena burutu: Azken ataza honetan dokumentuaren aurkezpen bat prestatuko da proeiktuaren xehetasun eta bereizgarritasun guztiak landuz.

2.2.3 Ordu banaketa

Behin LDE diagrama eginda dagoela eta lan paketeak ondo definiturik daudela, atal bakoitzari eskeiniko zaion ordu kopurua finkatu behar da. Jakinik GrAL-a 12 kredituko ataza bezela definitua dagoela, eta kreditu bakoitzerako 25 lan ordu esleitzen direla, nahiz eta lan hain pertsonala izan 300 orduko mugara ahalik eta estuen inguratu beharko da. 2.1 irudian ikus daiteke ordu banaketa.

Lan Paketea	Iraupena
Proeiktuaren kudeaketa	**20**
Plangintza	8
Arriskuak eta prebentzioak	2
Jarraipena eta kontrola	10
Ikerketa lana	**30**
I.I.-ri buruzko informazioa lortu	10
MLOps-ari buruzko info.-a lortu	10
Artearen egoera aztertu	10
Produktua	**170**
Lan modulua prestatu	40
Froga kasuak egin	40
Kasu erreala landu & ondorioztatu	70
Mikrozerbitzuekin martxan jarri	20
Dokumentazioa	**80**
GrAL-aren dokumetua burutu	70
Aurkezpena egin	10
Guztira	**300**

2.1 Taula: Proeiktuko lan paketeen ordu banaketa.

Produktua izango da gehien landu den atala, izan ere, hau da lan osoaren muina. Hala ere, dokumentuari ordu kopuru esanguratsua eskeiniko zaio eta Proiektuaren kudeaketa eta aurretiko ikerketa lana ere kontuan hartu beharko dira.

2.2.4 Emangarriak

Hauek dira proeiktuan zehar burutu behar izan diren emangarriak eta bakoitzak esleitua duen muga:

- Gradu Amaierako Lanaren **dokumentua**: 2023-06-25.
- Gradu Amaierako Lanaren **aurkezpena**: 2023-07-03.

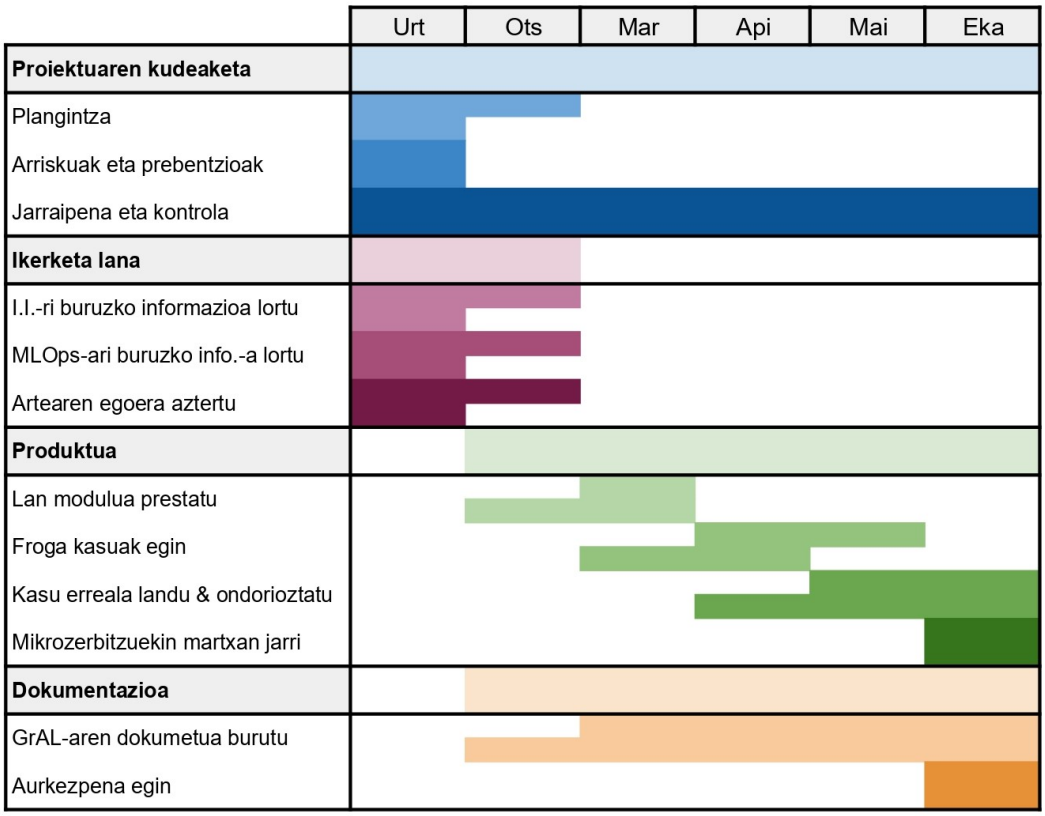
2.2.5 Lan metodologia eta bilerak

Proeiktua aurrera eramateko, lana bai Artelnics enpresan bertan, bai etxetik egina izan da, izan ere, beharrezkoa produktu guztiak edozein plataforman edo ordenagailuan eskuratu ahal izan ditut Artelnics-eko gitLab era konektatuz enpresan eskeinitako VPN-aren bitartez.

Bestetik, bilerak presentzialki, hala nola online egin dira, momentuko egoeraren eta prestasunaren arabera, izan ere, bi alderdiekiin egin behjjar izan ditut, bai enpresako tutorea eta bai fakultateko tutorea. Fakultateko tutorearekin egin dako bileren jarraipena 2 asterokoa izan da normalean (salbuespenak salbuespen), enpresarenak berriz ia egunerokoak.

2.2.6 Gantt diagrama

Azpiatal honetan Proiektuen Kudeaketa ikasgaian landu den erara, Gantt diagrama 2.2 irudian ikus daiteke.



2.2 Taula: Proiektuaren Gantt diagrama.

Ordu banaketaren atalean aipatu denez, produktuari eskainiko zaizkio ordu gehienak, baino denboran zehar gehien hedatuko den blokea jarraipena eta kontrola izango da. Proiektuaren fase orotan jarraipena burutuko da eta suertatutako arazo edo lorpenak dokumentatuak izango dira.

2.3 Arriskuak eta prebentzioak

Atal honetan, proiektuaren garapenean zehar egon daitezkeen arriskuak identifikatuko dira, eta bakoitza nola konpondu litekeen adieraziko da:

- **Engine-arekin arazoak izatea**

  - **Arazoaren deskribapena**: Ikasketa inkrementala burutu ahal izateko, Engine edo lan modulu bat sortzea da asmoa (aurretik aipatu den bezela). Baliteke Engine honen logika ondo ez inplementatua egotea eta emaitza txarrak lortzea, hau da, ikasketa inkrementala guztiz ondo ez funtzionatzea.
  - **Nola konpondu**: Hau gertatu ezkero, proiektuan atzera pausu bat egin beharko litzateke kasu errazago bat erabiliz ikasketa inkrementala frogatzeko. Hau egin ahal izateko, egiten den aldaketa bakoizteko *checkpoint* bat gordea izango da atzera pausuak era erraz batean eman ahal izateko.

- **VPN-a ez funztionatzea**

  - **Arazoaren deskribapena**: VPN honen bidez, Artelnics enpresak duen gitlab-era konektatu naiteke. VPN-ak ez funtzionatu ezkero, neural designerraren azken bertsioa ezin izango nuke eskuratu (jakinik lana neural designerrean oinarritzen dela).
  - **Nola konpondu**: Arazo honi irtenbidea eman ahal izateko, mail bidez Neural Designer-aren azken bertsioaren konprimatuak eskuratu ahal izango ditut.

- **Helburuetara ez iristea**

  - **Arazoaren deskribapena**: Baliteke prozesuan zehar helburuak gehigizkoak izan direla ikustea, eta lana mugetara iritsiko ez dela iksutea.
  - **Nola konpondu**: Hau gertatu ezkero bi konponbide daude. Ekainerako epera iritsiko ez naizela ikusi ezkero, bigarren deialdira, hau da, Iraileko deialdira apuntatzea besterik ez da geratuko. Hau egin eta gero, hala ere bigarren deialdi honen mugetara ez banaiz iristen, helburuak eguneratu behar izango dira, zentzuzko berri batzuk bilatzeko.

- **Mikrozerbitzuak ezin erabiltzea**

  - **Arazoaren deskribapena**: MLOps teknika aplikatzeko, kasu askotan Docker bezalako mikrozerbitzuak erabiltzen dira. Baliteke kasuren batean mikrozerbitzu hauek lan fluxuarekin batu ezin izatea.
  - **Nola konpondu**: Kasu honetan, helburu nagusiari garrantzi gehiago emango zaio eta mikrozerbitzuen erabilera alde batera utziko da (izan ere, hobekuntza bat besterik ez da).

2.4 Jarraipena eta kontrola

Atal honetan proiektuaren plangintza berrikusi egingo da, proiektua burutu den bitartean suerta diren arazoak eta plangintza aldaketak kontuan hartuz.

2.4.1 LDE diagrama eta Lan paketetak

LDE diagrama mantendu egin da eta ez da aldaketa nabaririk egon. Gainera, lan paketeen banaketa mantendu egin da eta ez da aldaketarik egon.

2.4.2 Emangarriak

Hauek dira proeiktuan zehar burutu behar izan diren emangarrien muga berriak:

- Gradu Amaierako Lanaren **dokumentua**: 2023-09-17
- Gradu Amaierako Lanaren **aurkezpena**: 2023-09-25 - 2023-09-29

2.4.3 Ordu banaketa

Atal honetan bereziki aldaketak garantzitsuak izan dira. Arriskuen arten jasota zegoen bezela, GrAL-a estimatutako denboran bukatzea zaila suertatu da, hortaz ordu banaketa berregin behar izan da hurrengo muga posiblera iristeko. Hortaz, 2.3 irudian ikus daiteke ordu banaketa berria.

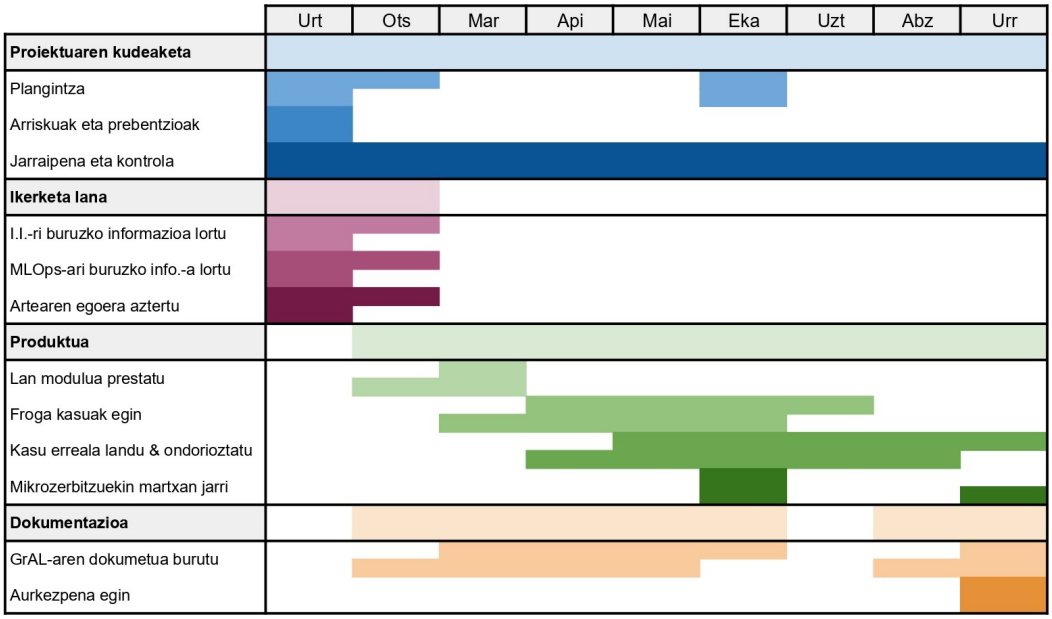
Lan Paketea	Iraupena
Proeiktuaren kudeaketa	**24**
Plangintza	10
Arriskuak eta prebentzioak	2
Jarraipena eta kontrola	12
Ikerketa lana	**30**
I.I.-ri buruzko informazioa lortu	10
MLOps-ari buruzko info.-a lortu	10
Artearen egoera aztertu	10
Produktua	**197**
Lan modulua prestatu	40
Froga kasuak egin	50
Kasu erreala landu & ondorioztatu	85
Mikrozerbitzuekin martxan jarri	22
Dokumentazioa	**90**
GrAL-aren dokumetua burutu	80
Aurkezpena egin	10
Guztira	**341**
Desbiderapena	**41**

2.3 Taula: Proiektuaren ordu banaketa finala.

Nahiz eta aurreikusita egon produktua izango zela ordu gehien beharko zuen atala, lanaren ordu banaketa eraldatu egin behar izan da bloke honi ordu gehiago eskeintzeko asmoz. Aldaketa txikiak burutu dira ere proiektuaren kudeaketaren atalean hala nola dokumentazioaren atalean.

2.4.4 Gantt diagrama

Azpiatal honetan Proiektuen Kudeaketa ikasgaian landu den erara, Gantt diagrama aurkeztuko da 2.4 irudian.



2.4 Taula: Proiektuaren Gantt diagrama finala.

Proiektua hedatu egin denez, jarraipena eta kontrola gogotsu mantendu da proiektua burutua izan den bitartean. Bestetik, bai produktu finalari, bai dokumentazioari eskeinitako ordu "extrak"irudikatu dira Gantt diagrama berri honetan.

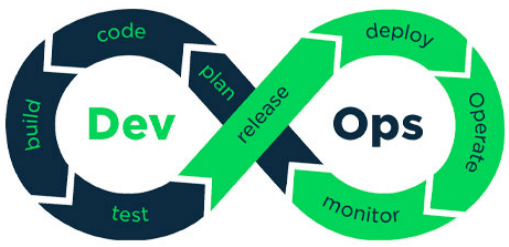
KAPITULUA 3

Kontzeptu teorikoak

Atal honetan lanean landuko diren kontzeptu teorikoak azaldu eta sakonduko dira. Lehenik, MLOps kontzeptua definitu egingo da, ondoren, ikaskuntza inkrementala zer den eta MLOps paradigmaren barruan nola funtzionatzen duen azalduko da, eta azkenik, bi gai hauen artearen egoera landuko da.

3.1 MLOPS

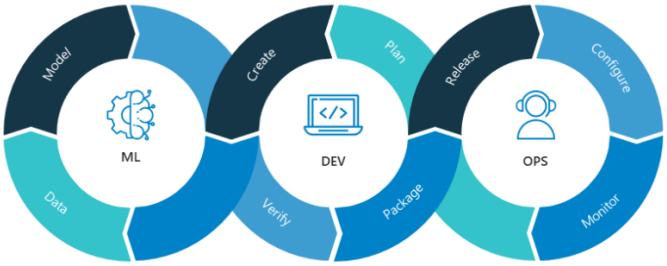
Orain dela gutxira arte, eskuz kudeatu genitzakeen gure enpresan eskura genituen datuak. Datuak maneiatzea edota prozesatzea ia artisau-prozesua zen, eta adituen esku uzten erakunde gehienetan. Hala ere, gaur egun datuetara bideratutako mundu batean gaude, geroz eta informazio-fluxu aldakorretaz betetako ingurune batean, gure datu-biltegiak etengabe betez eta osatuz. Egoera horri aurre egin behar izan zaio azken urteetan, eta horretarako, hainbat irtenbide agertuz joan dira, adibidez, Data Lake-ak sortuz, streaming bidezko datuen prozesatzeari aurre eginez eta enpresetan perfil hiper-espezializatuak berriak sortuz (hala nola ML Engineer edo Data Engineer-a). Hau guztiaz gainera, bada gure prozesuan kontuan hartu behar den beste funtsezko elementu bat: erabakiak IA bidez hartzea, hau da, erabakiak automatizatzea. Testuinguru honetan jaio ziren lehen produktuak, baina hauen bertsioak abiada azkarregian ekoizten hasi ziren produkzio paradigma tradizionalerako. Beraz, irtenbide gisa, DevOps-en praktikak sortu ziren. 3.1 irudian ikus daiteke ***DevOps*** [6] teknikaren lan fluxua.



3.1 Irudia: DevOps teknikaren lan fluxuaren diagrama.[1]

Egoera horrekin batera, MLren bizi-zikloen azterketak eta ***MLOps***-en praktikak (Machine Learning Operations) agertuz joan dira.

Era orokorrean eta iturri desberdinek definitzen duten bezala, **MLOps** edo **Machine Learning Operations**, **DevOps** metodologiaren hedapen bat da, ikaskuntza automatikoko eta datu konputazioko prozesuak garapen katean sartzea bilatzen duena, MLren garapena fidagarriagoa izan dadin eta produktuak ahalik eta goizago produzitu daitezen [7][8]. MLOPS teknikaren lan fluxua irudikatzen duen diagrama 3.2 irudia da.



3.2 Irudia: MLOPS teknikaren lan fluxuaren diagrama.[2]

Datuetara bideratutako mundu batean gaudenez, MLOps ereduek berebiziko garrantzia dute, izan ere, ezinbestekoak dira erakunde baten barruan IA eta ML proiektuen heltze prozesua arintzeko.

Entrenatutako ereduen erabilera erabat aldatu da azken hamarkadan, eta, orain, ML ereduak eragiketa tradizionaletan zituzten "kutxa beltz"gisa kudeatzeko politika ez da nahikoa. Gaur egun ereduak eguneratu eta mantendu egiten dira, hortaz, ezinbestekoa da hedatzeko eta mantentzeko ahaleginak eta denborak ahalik eta gehien murriztea, hala nola produktuaren monitorizazio etengabea izatea.

Beraz, hau guztia jakinik, zeintzuk dira **MLOps-aren abantailak**:

- Ereduen balioztapen fasean arriskua murriztea (erabilgarri izango ez diren ereduetan egindako inbertsioa murriztea)
- Soluzio konplexuagoen inplementazioa sinplifikatzea (modelo gehiagorekin, entrenamendu gehiagorekin, entrenamendu baldintzatuekin...)
- CI eta CD prozesuak, azkarrago eta errore gutxiagorekin (prozesu automatizatuen automatizazioari esker).
- Ereduak etengabe monitorizatzea eta eguneratzea, datuekin batera eboluzionatu dezaten (horrela IA sistemen eraginkortasuna hobetuz).

3.2 Ikasketa inkrementala

Ikaskuntza inkrementala, etengabeko ikaskuntza edo lineako ikaskuntza ikasketa automatikoaren arlo bat da, eta datu-fluxu batetik lortutako sarrerako datuak etengabe eta denbora

[1]https://www.rrhhdigital.com/secciones/tecnologia-e-innovacion/136859/
[2]https://la.blogs.nvidia.com/2020/09/08/que-es-mlops/

errealean prozesatzean datza [9]. Gainera, prozesu honetan ziurra da ezagutza urria edota hutsa izatea aldagai iragarleen banaketari buruz, laginaren tamainari buruz, iragarpenaren alderdiei buruz edo funtzio objektiboari buruz (doikuntza-parametroen balio egokiak barne) eta behaketek etiketak izateari buruz.

Ikasketa inkrementaleko algoritmoak malguak, eraginkorrak eta moldagarriak izan behar dute. Ezaugarri hauek dira ikaskuntza inkrementala eta ikaskuntza automatiko tradizionala bereizten dituztenak:

- Eredu inkremental bat datuetara azkar eta eraginkortasunez egokitzen da, horrek esan nahi du denbora errealean egokitu daitekeela datuen banaketan gertatzen diren aldaketeta edo desbiderapenetara.
- Behaketen etiketak dagozkion datu iragarleak ditugunean falta daitekenez, algoritmoak duen ereduaren azken bertsiotik abiatuta iragarpenak azkar sortzeko gai izan behar du, eta modeloaren entrenamendua atzeratu behar du ere.
- Litekeena da biztanleriari buruzko informazio gutxi ezagutzea ikasketa inkrementala hasi aurretik. Hortaz, algoritmoa hutsetik has daiteke. Adibidez, sailkapen arazoetarako, baliteke klaseen izenak ez ezagutzea ereduak oharrak prozesatu arte. Aldez, Ikaskuntza hasi aurretik informazio nahikoa ezaguna denean, informazio hori zehaztea posible izango da, ereduari hutsetik hasi ez dadin.

Ikaskuntza automatiko tradizionalean, etiketatutako datu-sorta bat dago balioztapen gurutzatua aurrera eramateko, orokortze-errorea balioesteko eta hiperparametroak doitzeko, aldagai iragarleen banaketa ondorioztatzeko eta eredua doitzeko. Hala ere, ateratzen den eredua hasieratik entrenatu behar da berriro, azpiko banaketak desbideratzen badira edo eredua degradatzen bada. Bestalde, ikaskuntza inkrementala erabiliz hiperparametroak doitzeko balioztapen gurutzatua egitea zaila bada ere, ikaskuntza inkrementalak malguak dira banaketaren desbiderapenera denbora errealean egokitu daitezkeelako. Horrela, modelo hauek era normalean (edo tradizionalean) entrenatutako ereduek dituzten predikzio zehaztasunari hurbilduko dira entrenatzen doazen heinean.

Demagun modelo inkremental bat prest daukagula honek iragarpenak sortzeko eta bere errendimendu prediktibo propioa neurtzeko. Ikaskuntza modelo inkremental honek sartzen diren behaketen paketeak emanik, datuak denbora errealean prozesatuko ditu, honako modu hauetakoren batean (normalean zehaztutako ordenan):

- **Eredua ebaluatzea**: Ereduaren errendimendu prediktiboaren jarraipena egitea benetako etiketak daudenean.
- **Desbideratzeak detektatzea**: Egiaztatu banaketan egitura hausturarik edo desbideratzerik dagoen. Adibidez, iragartzeko aldagairen baten banaketa nahikoa aldatu den zehaztea.
- **Modeloa berrentrenatzea**: Eredua eguneratu sartzen diren behaketekin entrenatuz (Benetako etiketak izanez gero edo egungo eredua behar adina degradatu baldin bada).
- **Iragarpenak sortzea**: Etiketak aurreikustea azken eredutik abiatuta.

Ikaskuntza inkrementalaren garrantzia, neurri handi batean, mundu osoan gertatzen ari den datuen eztanda esponentzialari zor zaio. Datuak geroz eta kopuru handiagoan daude eskuragarri, eta geroz eta gehiago denbora errealean, beraz, ikaskuntza automatikoko ereduek eta sare neuronalek etengabe egokitzeko eta eguneratzeko gai izan behar dutela esan nahi du honek horrela hauen eraginkortasuna bermatzeko.

Gainera, azken urteetan oso garrantzitsuak diren esparruetan, lengoaia naturalaren prozesamenduan [10] edota ahotsaren ezagutzan adibidez [11], etengabeko ikaskuntzak garrantzi handiko papera jokatzen du ereduak beraien eraginkortasuna mantentzeko gai izan behar baitira hizkuntzak dituen aldaketak jasanez eta pertsonek hitz egiteko duten erara egokituz. Jarri dezagun kasu praktiko bat:

Demagun **termostato adimendun** bat dugula. Honek, automatikoki tenperatura bat ezarriko du giro-tenperatura, hezetasun erlatiboa, eguneko ordua eta beste neurketa batzuk kontuan hartuta, eta gainera, erabiltzailearen barne-tenperaturaren lehentasunak ikasteko ahalmena izango du. Demagun ere fabrikatzaileak gailu hau prestatu duela eredu ezagun bat gehituz, neurketak kontuan hartuta batez besteko pertsona baten lehentasunak deskribatzen dituena. Instalatu ondoren, gailuak minuturo bilduko ditu datuak, eta tenperaturak bere balio lehenetsietara egokituko ditu. Termostatora hasieratik gehitutako modeloa ber-doituz joango da momenturo, erabiltzaileak gailuarekin dituen ekintzen edo inakzioen arabera patroi hauek ikasketa algoritmoaz kontuan izanik. Ziklo honek mugarik gabe jarraitzeko ahalmena izango du. Termostatoak datu historikoak gordetzeko diskoan espazio mugatua badu, denbora errealean berrentrenatu beharko da. Baliteke ere termostato honek hasieratik modelo ezagunik ez izatea. Kasu honetan, modeloak bere doikuntza propioak egin ditzan patroi hauek ikasteko, hasieratik modeloa sarriago entrenatzeko ahalegina egingo da.

3.3 Artearen egoera

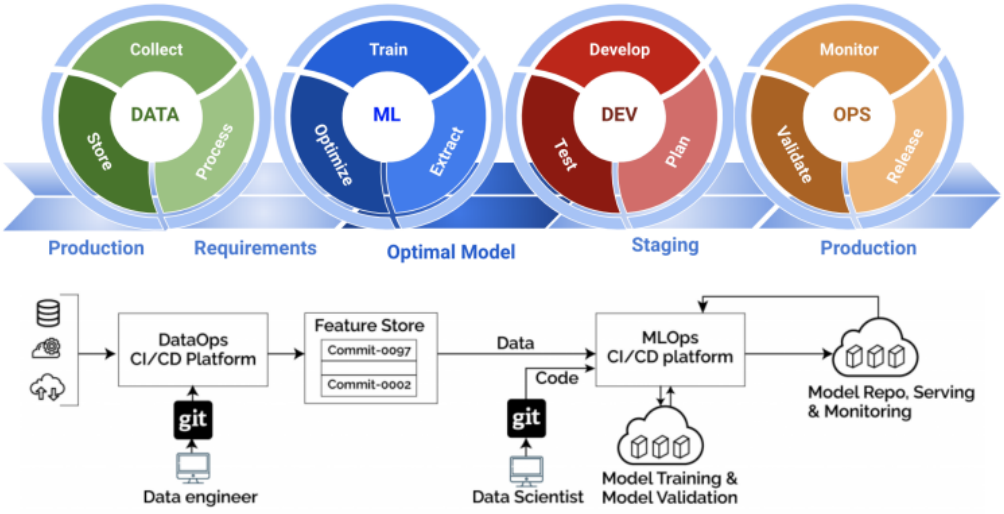
Lanarekin jarraitu aurretik, garrantzitsua da ulertzea gaur egungo egoeran lan hau non kokatzen den, eta batez ere Ikaskuntza inkrementalari gaur egun ematen zaizkion erabilerak ulertzea hala nola hauek MLOps paradigmaren barruan nola integratzen diren.

Gaur egun, ikaskuntza automatikoaren aurresateko ereduak modu egoera-gabean eta garestian eguneratzen dira, eta ikaskuntza automatikoan oinarritutako aplikazioak eta sistemak sortu nahi dituzten enpresek hautatutako bi joera nagusiak denbora errealeko inferentzia eta etengabeko eguneratzea dira. Horregatik **Continual-Learning-as-a-Service (CLaaS): On-Demand Efficient Adaptation of Predictive Models** [1] artikuluan, zerbitzuak eta software-ereduak mailegatzeko azpiegitura berritzaile bat definitzen da, *Continual-Learning-as-a-Service* edo **CLaaS** izenekoa ("*Etengabeko-Ikaskuntza-Zerbitzu-gisa*"euskeraz), arazo horiei aurre egiteko. Zerbitzu honek etengabeko ikaskuntza automatikoko eta etengabeko integrazioko teknikak hartzen ditu zehazki eta Datu-zientzialarientzako ereduak eguneratzeko eta baliozkotzeko tresnetarako euskarria eskaintzen du (modu eraginkorrean). Hau guztia aurrera eramateko, artikulu honek **Continual Brain** izeneko CLaaS instantziaren diseinua eta inplementazioa aurkezten du, benetako bi agertokitan ebaluatua.

Plataforma honen funtzionatzeko adierazten duen grafikoa 3.3 irudian ikus daiteke.

MLOps teknikaren irudikapen grafiko honek, ikaskuntza automatikoaren bizi-zikloaren kudeaketa integrala erakusten du. Prozesua interaktibo eta inkrementala da, batez ere 3 faseetan oinarritua: **Datu fasea**, **Machine Learning (ML) fasea** eta **DevOps fasea**.

Bigarren fasean bi printzipio garrantzitu bereizi ditzakegu: **Integrazio jarraitua (Continious Integration, CI)** eta **Etengabeko entrega (Continious Delivery, CD)**. Honetaz gain, MLOps-ek beste bi praktika gehitzen ditu: **Monitorizazio etengabea (Continious Monitoring, CM)** eta **Ikasketa etengabea(Continious Training, CT)**.



3.3 Irudia: MLOPS teknikaz gidatutako Ikaskuntza Automatikoaren bizi-zikloaren kudeaketa integrala [1]

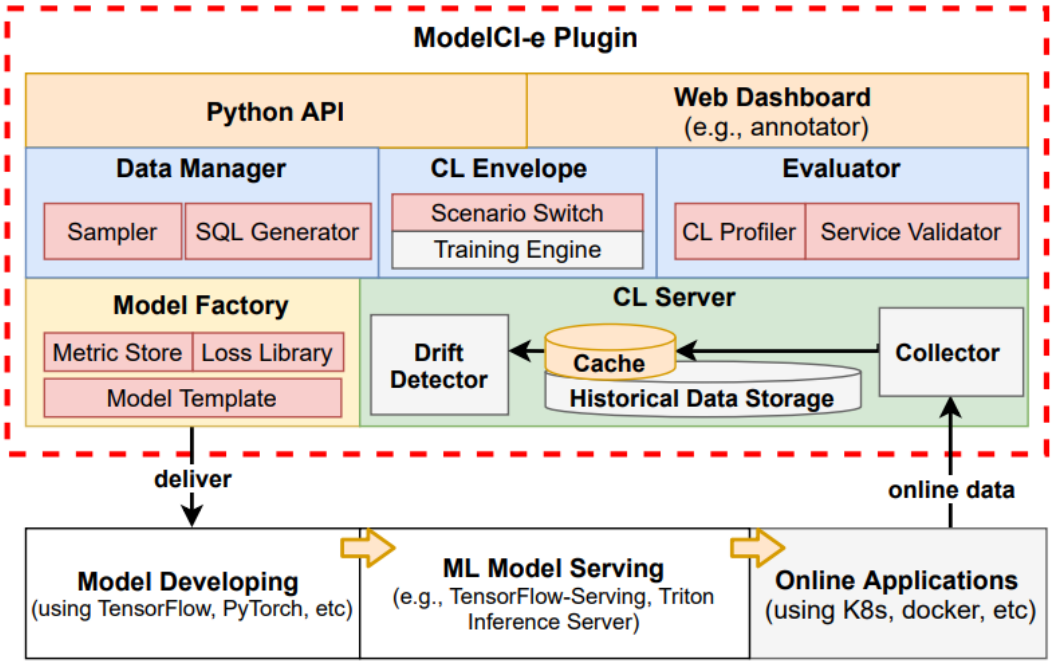
Artikulu honetatik lortutako informazioa, zehazki Grafiko honetan ikusitakoa, oso baliogarria da lan honen garapenerako, izan ere, inplementazio posible bat erakusten da MLOps paradigmaren barruan Etengabeko Ikasketa modeloak (lan honetan azterten ari den Ikasketa inkrementala bezalakoak) integratzeko. Ikusitako bizi zikloak, garrantzia handia izango du lan honetan aurrerago proposatuko den bizi zikloaren atalean.

Bestalde, Aurretik ikusi dugunez, MLOps *Machine Learning* (ML) eredu esperimentalak produkziora eramatean datza, hau da, ereduak benetako erabiltzaileei zerbitzatzean baino zoritxarrez gaur egun dauden ML zerbitzu sistemek ez dituzte behar bezala kudeatzen lineako datuak lineatik kanpoko entrenamendu datuetatik aldentzen diren ingurune dinamikoak, eta horrek ereduak eguneratzeko eta hedatzeko lan gogaikarriak eragiten ditu.

Horretarako, **ModelCI-e: Enabling Continual Learning in Deep Learning Serving Systems [2]** artikuluan ikusten den bezala, MLOps teknikarentzako osagarri arin bat inplementatzen da, ModelCI-e izenekoa (CI-e = *Continious Integration* and *evolution*, edo euskeraz, "Etengabeko Integrazioa"eta bilakaera), arazo horri heltzeko. Zehazki, etengabeko ikaskuntzako eta ML-eko erakuste teknikak hartzen ditu, horrela, ModelCI-e-k euskarri integrala ematen du modeloak eguneratzeko eta baliozkotzeko, motorra pertsonalizatu beharrik gabe. Hauek dira, besteak beste, ModelCI-e-k dituen ezaugarriak:

- Modelo generadore bat du. Horrela, *Continual Learning*-eko erreferentziazko prototipoak eta CL ereduak erraz sortzeko aukera du.
- *Continual Learning* backend atal bat du. Hau, modeloen eguneratzea modu eraginkorrean automatizatu eta orkestratzeko da.

- Web interfaze bat du, Machine Learning-eko talde batek CL zerbitzua elkarlanean eragozpenik gabe kudea dezan.



3.4 Irudia: ModelCI-e-ren arkitektura[2]

Ikerketa honetatik, garrantzitsua da batez ere ulertzea ModelCI-e-ren arkitetura nolakoa den, izan ere, lan honetan aurrera eramango den proiektuak funtziona dezan lan moduluak antzeko arkitektura bat izango duelako.

Atal bereizgarrienen artean datu tratamendua eta datu bistaraketa aipatuko nituzke, MLOps teknikan ikusi dugunez, berebiziko garrantzia duten atalak izanik ondo landuta egotea oso garrantzitsua baita, eta kasu honetan, bereziki landutako atalak direla ikusi daiteke.

Aipatzekoa da ere, lan honetan aurrera eramango den proiektuak honek egiten duenaren antzera emaitzak *dashboard* batean bistaratuko dituela. Bestetik, datu basea ez da SQL bidez tratatuko.

KAPITULUA 4

Teknologiak eta tresnak

Kapitulu honetan lanean zehar erabiliak izan diren tresna eta teknologiei buruz hitz egingo da, hauei buruzko xehetasunak adieraziz, eta hauen erabilera nolakoa izan den argi utziz.

4.1 Neural Designer

Hasteko laburki defini dezagun aurrerago definituko dugun lan modulua sortzeko ezinbestekoa izan den tresna.

Neural Designer, **Artelnics** taldeak sortutakoa, datu zientzi eta ikaskuntza automatikorako aplikazio efektibo, erraz eta intuitiboa da. IA-n oinarritutako aplikazioak sortzea ahalbidetzen du, bloke-diagramarik sortu edota programatu gabe.

Aplikazio honek ingurune grafiko txukun eta ulergarri bat izateaz gain, **OpenNN** liburutegia erabiltzen du azpitik (Artelnics taldeak sortutakoa ere) eta nahiz eta garapenean garrantzi handirik ez izan, ezinbestekoa da hau aipatzea Nerual Designer aplikazioan duen pisuagatik.

Gehiago sakontzeko, azaldu dezagun nolakoa den Sare neuronal ereduen tratamendua tresna honetan.

4.1.1 Sare Neuronalak Neural Designer aplikazioan

Aurretik esan den bezela, Neural Designer softwareak ereduak era oso erraz batean erabiltzeko aukera ematen du, izan ere, erabiltzaile ingurumenean 4 atal desberdintzen dira ereduen ulermena erraztuz:

- **Data Set**: Lehen atal honetan, datu baseko datu guztien artean *target* datuak, *input* datuak edo *unused* datuak zeintuzk diren aukeratuko dira. Beste era batera esanda datuen prozesamendua egingo da.

- **Neural Network**: Atal honetan sare neuronala bera editatzen da, honi kapak gehituz, neurona kopurua aldatuz edota aktibazio funtzioa aukeratuz.

- **Training Strategy**: Atal honetan sareak entrenatzeko erabiliko duen algortimoa hala nola beste hainbat parametro zehazten dira.
- **Model Selection**: Algoritmo honek orokortzeko gaitasun hoberena duen sarearen arkitektura neuronala bilatzen du.

4.1 Irudia: Lau atalen artean mugitzeko NeuralDesigner aplikazioak duen nabigazio barra

Programak datu hauek guztiak **NDP**, Neural Designer Proyect, fitxategi batean jasoko ditu (NDP fitxategi hau XML modukoa da) eta honek garrantzia du bereziki, izan ere, fitxategi hau izango da hurrengo ataletan definituko den lan moduluaren muina. Behin 4 ataletan geure aukerak egin ditugula, programak gauza asko egiten utziko digu, esaterako, sarrera dituen korrelazioak kalkulatu, datuak filtratu, sarearen parametroak ausaz ezarri, sarea entrenatu, behin entrenatua izan dela hainbat programazio-lengoaietara exportatu ...

Ikusi daitekenez programak funtzionalitate asko ditu, baina gaur egun agertzen diren arazo askoetan zerbait gehiago behar da, izan ere, datu kopuruak geroz eta handiagoak dira eta modeloak denbora errealen berregokitzen egon behar dute. Horrela bada, lan honetan Neural Designer aplikazioaren funtzionalitateak erabiliz, ikasketa inkrementalaren inplementazio bat porposatuko da.

4.2 Datu basea eta API-ak

Lan fluxuaren atalean datu baseen tratamenduari buruz gehiago hitz egingo den arren, ezinbestekoa da laburki hauek azaltzea eta hauen garrantzia aipatzea. Lan moduluak CSV artxiboak erabiliko ditu datu base bezala hauek atzituz entrenamenduaren ziklo bakoitzean. Zehazki, proiektura lotuta dagoen CSV-aren edukia aldatzen den bakoitzean lan moduluan dagoen modeloa eguneratu egingo da, sartu den informazio berri hau momentu honetatik aurrera kontuan izateko.

Bestalde aipatu beharra dago ere kasu errealean erabiliko den informazioa bi API desberdinetatik lortu dela (Aurrerago zehaztuko direnak). Bai API-en deiak bai Datu baseen tratamendua python bidez egin da.

4.3 Docker

Docker kode irekiko plataforma bat da, garatzaileei "**edukiontziak**"deituriko kapsulak sortzeko, hedatzeko, exekutatzeko, eguneratzeko eta kudeatzeko aukera ematen diena [12]. Edukiontziak elementu estandarizatu eta exekutagarriak dira, eta barruan daramaten aplikazioaren iturburu-kodea edozein ingurunetan kode hori exekutatzeko beharrezkoak diren sistema eragileko (SO) liburutegi eta dependentziekin konbinatzeko ahalmena dute. Era honetan, edukiontziek erraztu egiten dute aplikazio banatuen garapena eta entrega.

Container-en erabilera praktika geroz eta ezagunagoa bihurtu da, erakundeek ingurune hibridoetara eta hodeieko jatorrizko garapenera pasa ahala. Dockerrik gabeko edukiontziak sor daitezke, Linuxen eta beste sistema eragile batzuetan integratutako gaitasunekin

zuzenean lan eginez, baina Dockerrek edukiontziak azkarrago, errazago eta seguruago sortzen ditu.

Hau da aurrera eramango den lanean Docker integratzearen zergaitia.

4.4 Dash

Dash kode irekiko framework bat da, datuak bistaraketak ahalegintzen dituzten interfazeak sortzeko. 2017an abiarazia Python-en liburutegi gisa, urteak pasa ahala haziz joan egin da Dash, eta gaur egun R, F# eta Julia-rentzako inplementazioak aurki ditzazkegu. Dash-ek datuen zientzialariei web aplikazio analitikoak sortzen laguntzen die, web garapeneko ezagutza aurreratuen beharrik gabe.

Ezaugarri aipagarrienen artean, hiru teknologia dira Dash framework-aren muina:

- **Flask**: Web zerbitzariaren funtzionaltasuna ahalbidetzeko.
- **React.js**: Web orriaren erabiltzaile-interfazea renderizatu egiten du.
- **Plotly.js**: Aplikazioan bukaeran erakutsiko dituen grafikoak edota diagramak sortzen ditu.

4.2 Irudia: Dashboard aplikazio baten eredua, Dylan Castillo, 2023

KAPITULUA 5

Lan moduluaren garapenaren

Helburuetan aipatu denez, lan hau bi zati garrantzitsuetan banatu da. Alde batetik Ikasketa inkrementalaren azterketa dugu, baino hau burutu ahal izateko, lan modulu bat garatu da ikasketa prozesua ahalbidetzeko eta prozesu honetan izandako erroreak tratatzeko. Kapitulu hau lan modulu honetan zentratzen da, bereziki, moduluak dituen xehetasunetan eta hau garatzeko jarraitu den prozesuan.

5.1 Lan moduluaren diseinua

Ikasketa inkremenentalaren inplementazioa arazoz josita egon da prozesu osoan zehar, izan ere, prozesu hau aurrera eramateko prest ez dagoen software bat erabili da oinarri bezala. Hala ere, hainbat tekniketaz baliatuz, hau aurrera eramatea posible izan da. Gehiago sakontzeko, has gaitezen proiektuaren oinarria definitzen.

Lehenik azaldu beharra dago NeuralDesigner aplikazioak *NeuralEngine* exekutagarria erabiltzen duela beharrezko prozesuak aurrera eramateko. Prozesu bat edo beste jaurtitzeko, *Task* deituriko fitxategiak sortzen dira aplikazioan *NeuralEngine*-ari agindu zehatzak pasatzeko.

Hau jakinik, Ikasketa Inkrementala burutzeko diseinatua izan den lan moduluak (5.1) zentro bezala *NeuralEngine*-a izango du, eta ataza zehatzak burutuz joango da generatutako aginduen arabera. Hortaz MLOPS teknikaren behar guztiak betetzeko beharrezkoak diren atalak (hala nola datuen prozesamendua eta prozesuaren monitorizazioa) lan modulutik kanpo egingo da.

NeuralDesigner aplikazioan proiektu bat sortzerakoan ***.ndp*** luzapenaz identifika daiteken fitxategi berri bat sortzen da non proiektuaren informazio guztia gordetzen den (fitxategi hau XML motakoa da). Ondorioz, programan proiektu bat irekiz gero eta honetatik zehaztasunen bat aldatzen bada (esaterako neuronak dituen perzeptroi kapa kopurua), aldaketa hau aplikazioan jasotzeaz gain ***.ndp*** fitxategian jasoko da. Hortaz, lan moduluaren logika ***.ndp*** fitxategietan gordetako ereduak atzitzean eta eraldatzean datza.

Gehiago zehazteko, hauek dira proiektu berri bat sortzen denetik entrenamendu prozesua burutzen den arte ematen diren pasuak:

- Proiektua sortzen denean XML-a hutsik sortzen da, eta honi ***.csv*** fitxategi bat esleitzen zaionean (datu basea), XML-a eraldatu egiten da datu instantziei errferentzia egiteko. Prozesu hau errepikatzen baldin bada (hau da, beste datu base berri bat bat edota dagoena bera berriro irekitzen baldin bada) XML artxiboa borratu eta berridatzi egiten da.

- Ondoren, entrenamendu prozesua aurrera eramateko, XML-a irakurtzen da hemen dauden jarraibideak ezagutzeko (Sare neuronalaren xehetasunak, entrenamenduaren ezaugarriak...). Ondorioz, sare neuronalak entrenamendua burutzea nahi baldin badugu, parametro hauen arabera era bateko edo besteko jokabidea izango du. Gainera prozesu honek output textua generatzen du, geroago garrantzitsua izango dena.

Hau jakinik, pauso hauen tartean *breakpoint* moduko bat jartzea beharrezkoa izan dela ikusi da, modeloaren *checkpoint* bat gordetzeko, eta entrenamendua hasi baino lehen hau kargatzeko. Modu honetan prozesuak era honetako eskema izango luke:

- Proiektua sortu (bere XML-arekin batera) eta honi datu base bat esleitu, zehazki, ***.csv*** fitxategi bat.

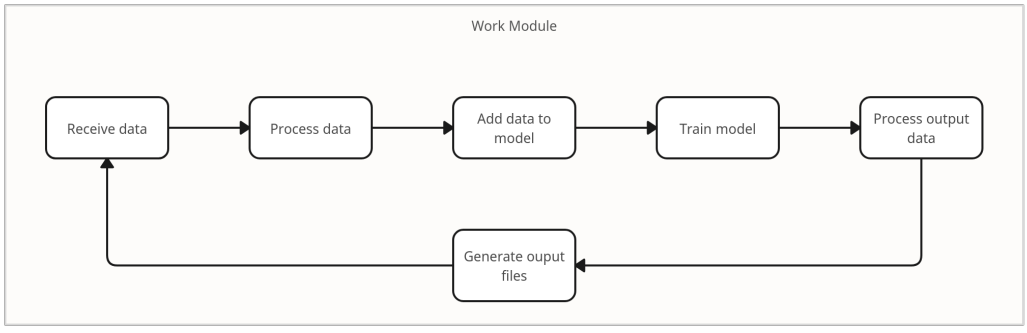
- Ondoren, aldaketarik egin nahi ez bada, lehen entrenamendu prozesua aurrera eramango. Entrenamendua ondo joan bada modeloaren *checkpoint* bat gordeko da.

- *Checkpoint* hau entrenamendu aurreko modeloari kargatuko zaio, horrela entrenamendu honekin lortu diren hobekuntzak jasoz.

- Era berean, entrenamenduak sorrarazi dituen fitxategi guztien artetik, *output*-en prozesamendua egingo da, zehazki hiru gauza lortzeko: Modeloa python lengoaira itzulia, entrenamenduaren output-en testu fitxategia, eta entrenamenduaren errore grafikoa.

- Berriz ere, datuak prest badaude, datu berriak gehituko dira datu basera.

- Datu berriak gehituak izan baldin badira entrenamendua berriro burutuko da.



5.1 Irudia: Lan moduluaren lan fluxuaren eskema orokorra

5.2 Modulua docker teknologiarekin integratuz

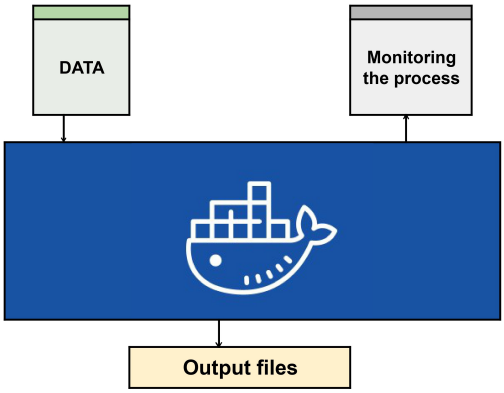
Lan moduluaren funtzionamendua definiturik dugula, hobekuntzetaz has gaitezke hitz egiten. Aurreko ataletan ikusi denez, Ikasketa Inkrementaleko prozesuak gaur egun biziki loturik daude MLOps terminoarekin, eta jakinik azken hau mikrozerbitzuak ondo integratzen dela (garapen-taldeen eta eragiketen arteko lankidetza eta komunikazioa sustatzen baitute), ezinbestekoa da hauetaz hitz egitea.

Mikrozerbitzuetan oinarritutako aplikazioentzako, ezinbestekoa da aplikazioak inplementatzeko unitate ideal bat eta exekuzio-ingurune autonomo bat edukitzea, eta hau da zehazki "kontainer"batek ematen duena. Kontainer-ak Docker bezalako tresnek kudeatzen dituzten kapsulak dira, eta kontainer edo kapsula bat definizioz, aplikazio baten kode eta mendekotasun guztiak formatu estandar batean paketatzen dituen formatua da, informatika-inguruneetan azkar eta fidagarri exekutatzeko aukera ematen duena.

Horregatik, lan moduluaren hobekuntza bezela edota proiektu honen etorkizunera begira, lan modulua *docker*-izatu edo enkapsulatu egin da, era honetan proiektu honi autonomia emanez eta exekuzio inguruneari buruzko kezkak deuseztatuz.

Labur esanik eta 4.3 atalean aipatzen den informazioa osatzeko, *docker* kontainer bat osatzeko, *dockerfile* deritzon fitxategi bat idazten da [13]. Fitxategi honetan, kontainerrak izango duen sistema eragilea, izango dituen aplikazioak, edo kontainerra pizterako garaian exekutatuko dituen aginduak definitzen dira.

Gainera, *dockerfile*-an, *docker* kontainerrak ordenagailutik hartu beharko dituen fitxategiak ere definitu daitezke. Honi esker, proiektua bukatuta dagoenean, kontainer bat sortu, *dokcerfile*-an proiektuak behar dituen dependentzia guztiak instalatzeko agindua idatzi, eta lan modulua ordenagailutik *docker*-era pasatzeko agindua definituko da. Era honetan lan modulua *docker* teknologia jasaten duen edozein makinan eskuragarri izango dugu.

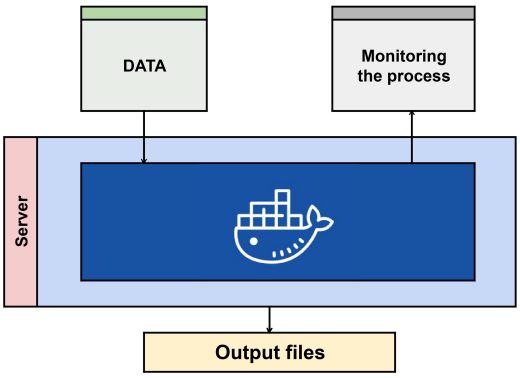


5.2 Irudia: LDE: Lan Deskonposaketa Eredua

Bestalde, aipatu beharra dago ere, lan modulu honek zerbitzarietan funtzionatzeko eragozpenik ez duela [14], lehen esan den bezala, *docker* teknologia jasaten duen edozein makinak lan modulu zehatz hau (edo beste edozein programa) duen kontainer bat exekutatu baitezake.

Horregatik, *docker* teknologia zerbitzariekin konbinatuz, zerbitzari aldean egokitutko eta etengabe ikasketen ari den kontainer bat presta liteke, adibidez. Honek, ikasketa

inkrementala burutzeko, eta modeloak etengabe entrenatzeko aukera oso ona sortzen du, eta proiektuaren etorkizunerako baliabide oso baliotsua izan liteke.



5.3 Irudia: LDE: Lan Deskonposaketa Eredua

KAPITULUA 6

Balidazio frogak

Behin implementazioa burututa zegoela, hurrengo pausoa honen funtzionamendua frogatzea izan da. Inplementazioa guztiz balidatzeko, 2 froga burutu dira. Lehenik sinu funtzioa aproximatuz, eta ondoren esponentzial funtzioa aproximatuz. Modu honetan, lan moduluak funtzionamentu egokia duela eta emaitzak zentzudunak direla frogatu nahi izan da. Entrenamendua eta emaitzak ikusi baino lehen, hurrengo ataletan funtzio bakoitzaren definizioak eta ezaugarri edo berezitasunak aurkeztuko dira, gainera, froga hauetan erabilitako modeloaren xehetasunak aztertuko dira.

6.1 sen(x) funtzioa

Matematikan, sinua sei funtzio trigonometrikoetako bat da, edo beste era batera, funtzio zirkularrak ere deituak. Funtzio erreal eta bakoitia da, $\mathbb{R}$ domeinua duena (zenbaki errealen multzoa) eta zeinaren kodomeinua $[-1, 1]$ tarte itxia den:

$$\text{sen} : \mathbb{R} \longrightarrow [-1, 1],\ x \longmapsto \text{sen}(x)$$

Trigonometrikoki, triangelu angeluzuzen baten angelu baten sinua, angelu horren kontrako katetoaren eta hipotenusaren arteko zatiketa gisa definitzen da no a katetoa den, eta c hipotenusa:

$$\sin \alpha = \frac{a}{c}$$

Funtzio sinusoidala ikasketa inkrementalarekin frogak egiteko ona dela esaten dugu, hasteko, funtzio hau periodikoa delako eta horrek esan nahi du baliagarria izan daitekeela portaera ziklikoak edo urtarokoak dituzten prozesuak modelatzeko. Kasu honetan, funtzio sinusoidala aukera ona da ikasketa inkrementala frogatzeko, izan ere, aukera ematen baitu datu berriak gehitu ahala neurona-sareak aldakuntza-ereduak atzemateko.

Bestalde baliagarria da ere portaera oszilakorrak edo harmonikoak dituzten prozesuak modelatzeko, pendulu baten mugimenduak kasu, eta horregatik, funtzio sinusoidala aukera ona da sareak portaera harmonikoak dituzten ereduak atzematen dakien edo ez frogatzeko.

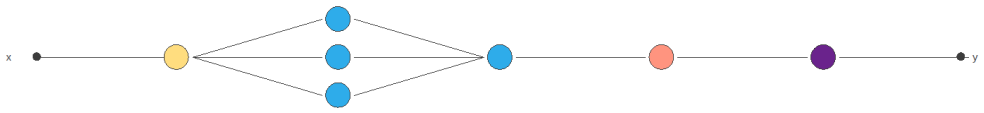
Azkenik, funtzio leuna eta etengabe bereizia dela dakigu, bereaz, egokia da optimizazio-algoritmoetan erabiltzeko, hala nola gradientearen jaitsieran, sare neuronalen entrenamenduan erabiltzen baitira.

6.1.1 Erabilitako Modeloa

Atal honetan sinuaren aproximazioa egiteko erabilia izan den modeloa azalduko da. Alde batetik honen arkitektura aztertuko da, eta ondoren, honen entrenamenduaren xehetasunak.

6.1.1.1 Sare neuronalaren arkitektura

Sarearen arkitektura 6.1 diagraman irudikatzen da. Irudi honen bidez sareak dituen geruza desberdinak erakutsi nahi dira, eta hauek hurrengoak dira: Geruza eskalarra neurona batekin (horia). Prezeptron geruza (3 neuronarekin) (urdina). Prezeptron geruza (neurona batekin) (urdina). *Unscaling* geruza (neurona batekin) (gorria). *Bounding* geruza (neurona batekin) (morea).



6.1 Irudia: Sare neuronalaren arkitekturaren diagrama.

Arkitektura hau defektuz Neural Designer aplikazioak eskuragarri jartzen duena da, eta aurrerago ikusiko denez, bai balidazio froga honetan, bai hurrengoan erabili da bi frogak baldintza berfinetan egiteko asmoz.

6.1.1.2 Entrenamendu xehetasunak

Atal honetan bi froga hauek aurrera eramateko orduan erabili den modeloa azalduko da. Argi ez badago, sare neuronal berdina hala nola entrenamendu xehetasun berdinak erabili dira bi kasuetan. Hasteko, galera indizea zehaztuko da, eta gero, erabilitako optimizazio algoritmoa.

Loss index: Galera-indizeak sare neuronalak egin beharreko zeregina definitzen du, eta ikasteko beharrezkoa den irudikapenaren kalitatea neurtzen du. Galera-indize bat ezartzean, bi kontzeptu hartu behar dira kontuan: **errorea** eta **erregularizazioa**.

Errore-terminoak neurona-sarea datu-multzora nola egokitzen den ebaluatzen du kuantitatiboki. Hainbat errore-metodo daude, eta egokiena aukeratzea aplikazio zehatzaren araberakoa da. Kasu honetan, Normalized Square Error, **NSE** (edo akats koadratiko normalizatua euskeraz), hautatua izan da. Akats koadratiko normalizatuak bateko balioa du sare neuronaleko irteerak helburuko aldagaien batez besteko balioen berdinak direnean; zeroko balioak, berriz, datuen iragarpen perfektua esan nahi du.

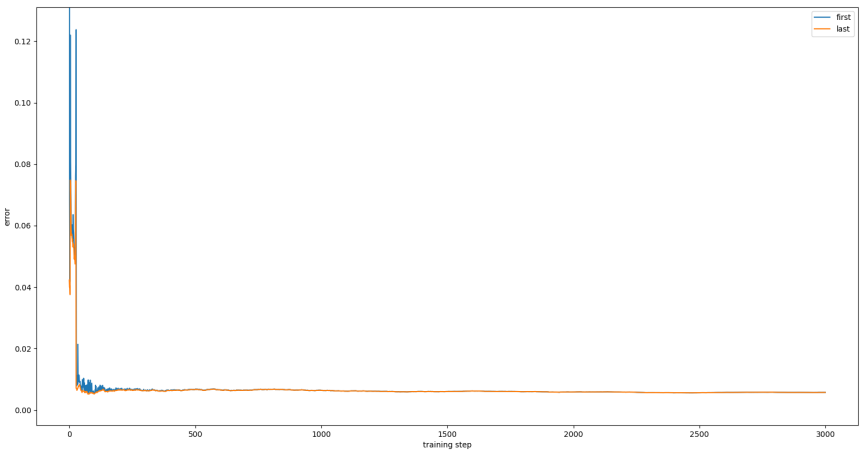
Erregularizazio-terminoak neurona-sarearen parametroen balioak neurtzen ditu. Erroreari gehitzen bazaio, neurona-sareak pisu eta alborapen txikiagoak izango ditu, eta, horrela, erantzuna leunagoa izango da eta gaindoikuntza saihestuko da. Kasu honetan, L2

erregularizazio-metodoa aplikatzen dugu. Neurona-sarearen parametro guztiak laukiari batuz lortzen da.

Optimizazio algoritmoa: Optimizazio-algoritmo gisa **cuasi-Newton** metodoa erabiltzen da. Newton-en metodoan oinarritzen da, baina ez du bigarren deribatuen kalkulurik behar. Horren ordez, cuasi-Newton metodoak algoritmoaren iterazio bakoitzean alderantzizko hessianoaren hurbilketa bat kalkulatzen du gradientearen informazioa soilik erabiliz.

6.1.2 sin(x) funtzioaren aproximazioa

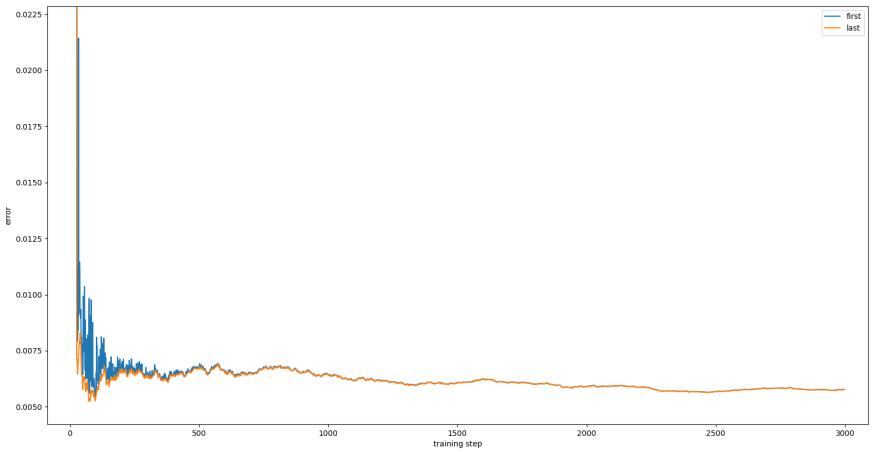
Atal honetan bi modelo konparatuko dira, hasierakoa, hau da, hasierako datu basearekin entrenatua izan dena, eta ikasketa inkrementalaren prozesua jaso egin duena. Has gaitezen entrenamenduaren erroreak aztertuz. Cuasi-Newton metodoaren erroreen grafika 6.2 eta 6.3 irudietan ikus daiteke. Pausu bakoitzeko, entrenamendu bat irudikatuko da, lortutako lehen (urdinez) eta azken (laranjaz) Cuasi-Newton erroreak grafikatuz. Hortaz marra urdinak eta laranjak 3000 entrenamenduz, hau da, 3000 puntuz osatutako marrak izango dira.



6.2 Irudia: Cuasi-Newton metodoaren erroreen grafika.

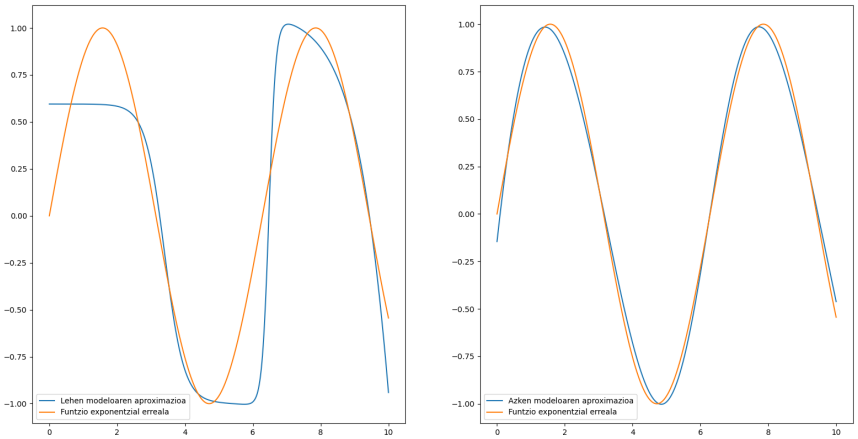
Ikusten denez, lehen 50 urratsetan tontor oso pronuntziatuak antzeman daitezke, eta hau, eredua datu-multzo txiki eta zaratatsu batera doitzen ari delaren seinale da. Datu-multzo txiki batekin eredu bat entrenatzen denean eredua datuetara gehiegi egokitzen da, daturik ez dagoen eskualdeetan doikuntza eskasa sortaraziz. Hala ere, entrenamendu multzoari datu gehiago sartu ahala, ereduak informazio gehiago lortuko du ikasteko, eta era berean, datu berrietara hobeto orokortzeko. Horregatik, entrenamendu-multzoari datu gehiago gehitzen zaizkion heinean, ereduak funtzio sinusoidalaren azpiko egiturak atzemateko gaitasuna handituz joango da, eta gutxinaka gutxinaka errore grafikoko tontorrak ez dira hain nabarmenak izango.

Bestetik, grafiko hau berezia ez dela esan daiteke, izan ere, normala da errorea pixkanaka gutxitu eta leuntzea, datu gehiago dituen eredu bat entrenatzen den heinean erroreen grafikoa denborarekin leunagoa eta errore txikiagokoa bihurtuko baitu.



6.3 Irudia: Cuasi-Newton metodoaren erroreen grafika lehen kurban zoom eginez.

Behin errore funtzioa analizatu dugula, 6.4 irudian ikus daiteken aproximazio grafikoa aztertuko ditugu. Kasu honetan bi grafiko desberdin izango ditugu, eta bietan, konparazio bat ikusiko dugu. Grafiko hauetan bi lerro azalduko dira. Alde batetik, lerro laranjak bai ezkerrean bai eskubian, eredu funtzioak (kasu honetan sinusoidala) izan beharko lukeen patroia irudikatu du. Bestalde, lerro urdinak, eredu bakoitzak egindako funtzio sinusoidalaren aproximazioa erakutsiko du. Ezkerreko lerroak, 26 daturekin egindako entrenamenduk lortutako aproximazioa irudikatuko du, bestalde eskubikoak, 3026 daturekin lortutako aproximazioa irudikatuko du.



6.4 Irudia: *sin(x)* funtzioaren aproximazioa.

Ezkerrean, modelo baseak egindako predikzioa ikusi dezakegu eta eskubian berriz, ikasketa inkrementalez entrenatutako modeloa. Desberdintasuna oso nabaria da. Lehen ereduak lortutako emaitzak ez du zerikusirik azken ereduak lortutakoarekin. 26 daturekin besterik ez entrenatua izan den eredua sinuaren funtzioaren egiturara oso gutxi gerturatu

da. Aldiz, 3000 pauso eta gero lortutako aproximazioa zerikusirik ez du. Kasu honetan, nahiz eta desfase nabarmena antzematen den funtzio errealaren eta aproximatutako funtzioaren artean, sare neuronalak sinuaren patroia guztiz ikasi duela ikus daiteke.

6.2 exp(x) funtzioa

Matematikan, funtzio esponentzial bat $f(x) = ab^x$ formako funtzio bat da, eta x berretzaile gisa aurkezten da. $f(x) = abc^x + d$ formako funtzio bat ere funtzio esponentziala da, honela berridatzi baitaiteke: $ab \cdot cx + d = (ab \cdot d)(bc)x$

Aldagai erreal bateko funtzio gisa, funtzio esponentzialen ezaugarri bakarra funtzio hauen hazkunde-tasa (hau da, deribatua) funtzioaren balioarekiko zuzenki proportzionala izatea da. Erlazio horren proportzionaltasun-konstantea b da, non: $\frac{d}{dx}b^x = b^x \log_e b$. Hortaz, $e = 2.71828$ konstantea da proportzionaltasun-konstantea 1 duen oinarri bakarra, eta horregatik, funtzioaren deribatua funtzio bera da: $\frac{d}{dx}e^x = e^x \log_e e = e^x$.

Funtzio esponentzialaren oinarriaren aldaketak faktore konstante gehigarri baten agerpena baino ez duenez lortzen emaitza gisa, konputazionalki komenigarria da funtzio esponentzialen azterketa funtzio partikular honen azterketara murriztea, konbentzioz "funtzio esponentzial naturala"edota "funtzio esponentziala"soilik deritzona, eta $x \mapsto e^x$ edo $x \mapsto \exp(x)$. notazioaz ezagutzen dena.

$\exp : \mathbb{R} \to \mathbb{R}$ funtzio esponentzial erreala hainbat modu baliokidetan adierazi daiteke. Hauen artean ezagunenak, "potentzia serie"edo "potentzia batura"erako definizioa eta limite bidezko definizioak dira hurrenez hurren:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

$$e^x = \lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n$$

Hortaz, definizioa ezagutuz, funtzio esponentziala *continious training*-aren edo ikasketa inkrementalaren inplementazioa frogatzeko aukera ona dela esan daiteke, batez ere baliagarria suerta daitekelako oraingo balioaren araberako kanbio-tasa duten prozesuak modelatzeko (esaterako biztanleria baten hazkunde esponentziala edo material erradioaktibo baten deskonposizioa). Kasu honetan, funtzio esponentziala aukera ona da continious training ezartzeko, sare neuronala balio-aldaketetara doitzeko aukera ematen baitu, datu berriak gehitu ahala.

Gainera, funtzio esponentziala saturazio-portaera duten prozesuak modelatzeko ere baliagarria izan daiteke, non aldaketa-tasak behera egiten duen balioa goiko edo beheko muga batera hurbiltzen den heinean. Kasu honetan, funtzio esponentziala aukera ona da ikasketa inkrementalaren inplementazioa probatzeko ere, sare neuronalak saturazio-portaera atzematea ahalbidetzen baitu datu berriak gehitu ahala.

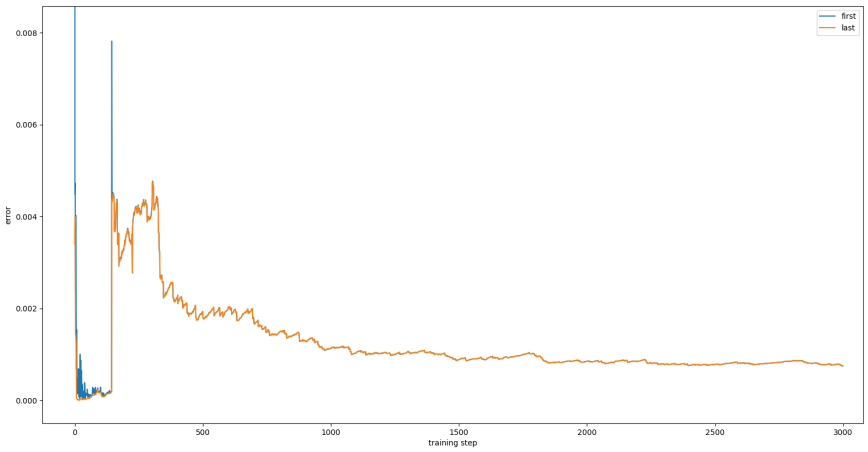
Bukatzeko eta sinuarekin gertatzen den bezala, esan dezakegu ere funtzio leuna eta etengabe bereizia dela, eta, beraz, egokia dela optimizazio-algoritmoetan erabiltzeko, hala nola gradientearen jaitsieran, sare neuronalen entrenamenduan erabiltzen baitira.

6.2.1 Erabilitako Modeloa

Kasu honetarako 6.1.1 atalean adierazitako erdeu berdina erabili da, bai arkitektura mantenduz, baita ere entrenamenduaaren xehetasunak mantenduz.

6.2.2 exp(x) funtzioaren aproximazioa

Funtzio sinusoidalarekin bezala, atal honetan bi gauza azalduko dira. Alde batetik, sare neuronalak entrenamenduan zehar izan duen entrenamendu errorea, esta bestetik, lehen modeloaren aproximazioa azken modeloaren aproximazioarekin aldaratuko da. Entrenamenduaren erroreen grafika 6.5 irudian ikus daiteke. Aurreko balidazio frogan bezala, pausu bakoitzeko, entrenamendu bat irudikatuko da, lortutako lehen (urdinez) eta azken (laranjaz) Cuasi-Newton erroreak grafikatuz.



6.5 Irudia: Cuasi-Newton metodoaren erroreen grafika.

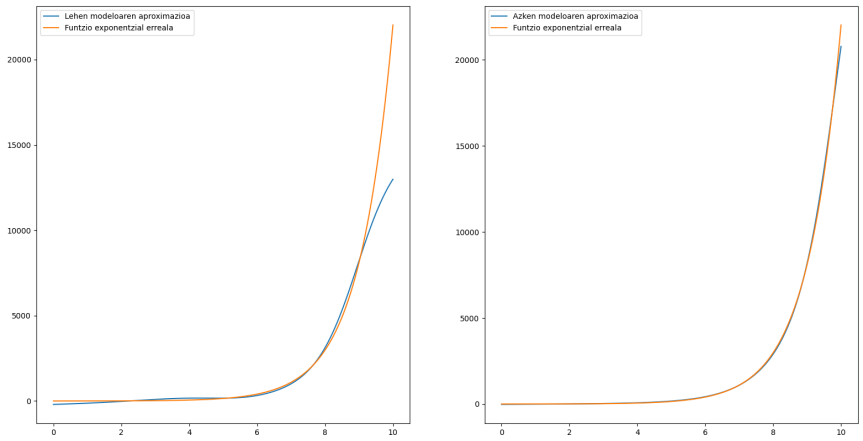
Lehenik eta behin has gaitezten entrenamenduaren lehen zatiari buruz hitz egiten, lehen 200 pausoei buruz zehazki. Hasieran ikus daiteke errorea oso azkar jaitsi egin dela, hau normala izanik noski, izan ere modeloak informaziorik ez izatetik 26 datu berri izatera pasatzen da, non patroian eta ereduak bilatu ahal izango dituen. Ikusten denez, 200 pausoetan zehar, errorea egonkortu egiten da berriro ere gorakada bat egiten duen arte. Tarte honetan, ereduak entrenatzen jarraitu eta datu gehiago jaso dituen heinean, litekeena da entrenamendu-datuak buruz ikasten hasi izana, sarrerako eta irteerako datuen arteko erlazioak orokortu beharrean, eta honek, ereduen errorea areagotu dezake entrenamenduan ikusi ez ziren datu berriekin.

Hau da espreski grafikoan ikusten den tontorrak esan nahi duena; eredua entrenamendu-datuetara gehiegi egokitzen ikasi du, eta horrek datu berrien aurrean errorea handitzea ekarri du. Fenomeno honi *Overfitting* deritzo. Tontor honetatik aurrera, eredua gutxinaka gutxinaka errorea murriztuz joan da, ikasketa inkrementalak erakutsi beharko lukeen errore patroi logikoa jarraituz, hau da, 0rekiko asisntotikoa den eta expnentzialki inbersoa den funtzioaren antza hartuz.

Aurreko atalaren antzera, 6.6 irudian ikus daitezken aproximazio grafikoak aztertuko

ditugu. Grafiko hauek bi lerroen arteko konparatiba bat irudikatuko dute, eta bakoitzaren esanahia hurrengoa da:

- Lerro laranjak, (aurreko kasuan bezela) bai ezkerrean bai eskubian eredu funtzioak izan beharko lukeen patroia irudikatu du, kasu honetan, exponentziala.
- Bestalde, lerro urdinak, eredu bakoitzak egindako funtzio exponentzialaren aproximazioa erakutsiko du.



6.6 Irudia: *exp(x)* funtzioaren aproximazioa.

Ezkerrean, modelo baseak egindako predikzioa ikusi dezakegu eta eskubian berriz, ikasketa inkrementalez entrenatutako modeloa. Konparazio honi esker ikus daiteke bigarren modeloak lortutako aproximazioa askoz zehatzagoa dela, batez ere, [8, 10] tartean, izan ere, lehen ereduak tarte honetan dituen emaitzak 5000 unitate azpitik daude. Dudarik balego modelo zehatzena aukeratzeari buruz (Aurreko kasuko desberdintasuna askoz nabariagoa izan baita), bi modeloak erreferentziarekin duten area kalkulatzea besterik ez genuke egin beharko, eta argi ikusiko litzateke bigarren modeloaren zehaztasuna lehenarekiko konparatuz.

KAPITULUA 7

Froga esperimentala

Ikusirik aurreko atalatean erakutsitako frogek erantzun zentzuduna erakusten zutela, aurrera beste pauso bat eman nahi izan da. Helburua atal honetan ez da izan emaitza zuzenak edo ez diren ikustea (lan hau dagoeneko eginda dago lehen frogetan), kasu honetan eszenario erreal bat aukeratu da eta ikasketa inkrementalak honen aurrean nola erantzuten duen aztertu da.

Ikerketa sendo bat egiteko asmoz, aukeratutako kasu erreala Madrileko kontaminazioa aurreikustea izan da, zehazki PM2.5, PM10, O3, NO2, SO2 kontaminazio parametroak. Kapitulu honetan, entrenamendua erakutsi, eta honen emaitzak eztabaidatu egingo dira, baino hau egin aurretik datu basea eta modeloa azalduak izango dira.

7.1 Modeloa

Modeloa aztertzeko bi atal garrantzitsuetan sakonduko dugu: Entrenamenduko xehetasunetan eta erabilitako arkitekutan.

7.1.1 Entrenamenduko xehetasunak

Atal honetan bi froga hauek aurrera eramateko orduan erabili den modeloa azalduko da. Argi ez badago, sare neruonal berdina hala nola entrenamendu xehetasun berdinak erabili dira bi kasuetan. Dagoeneko 6.1.1 atalean definitu denez, galera indizearentzako bi kontzeptu hartu behar dira kontuan: **errorea** eta **erregularizazioa**.

Kasu honetan, **Batez Besteko Errore Koadratikoa** hautatu dugu. Akats Koadratikoaren batuketan oinarritzen da, abantaila bezela izanik bere balioa lagin kopuruarekin hazten ez dela. Erregularizazio dagokionez, L2 erregularizazio-metodoa aplikatuko da (hau, neurona-sarearen parametro guztiak laukiari batuz lortzen da).

Bestalde optimizazio algoritmoari dagokionez, *Moment Adaptative Estimation Method* edo *ADAM* algoritmoa (Uneen kalkulu egokitzailearen metodoa euskeraz) izan da erabili dena [15].

7.1.2 Sare neuronalaren arkitektura

7.1 Irudian ikus daiteke erabilitako sare neuronalaren irudikapena.

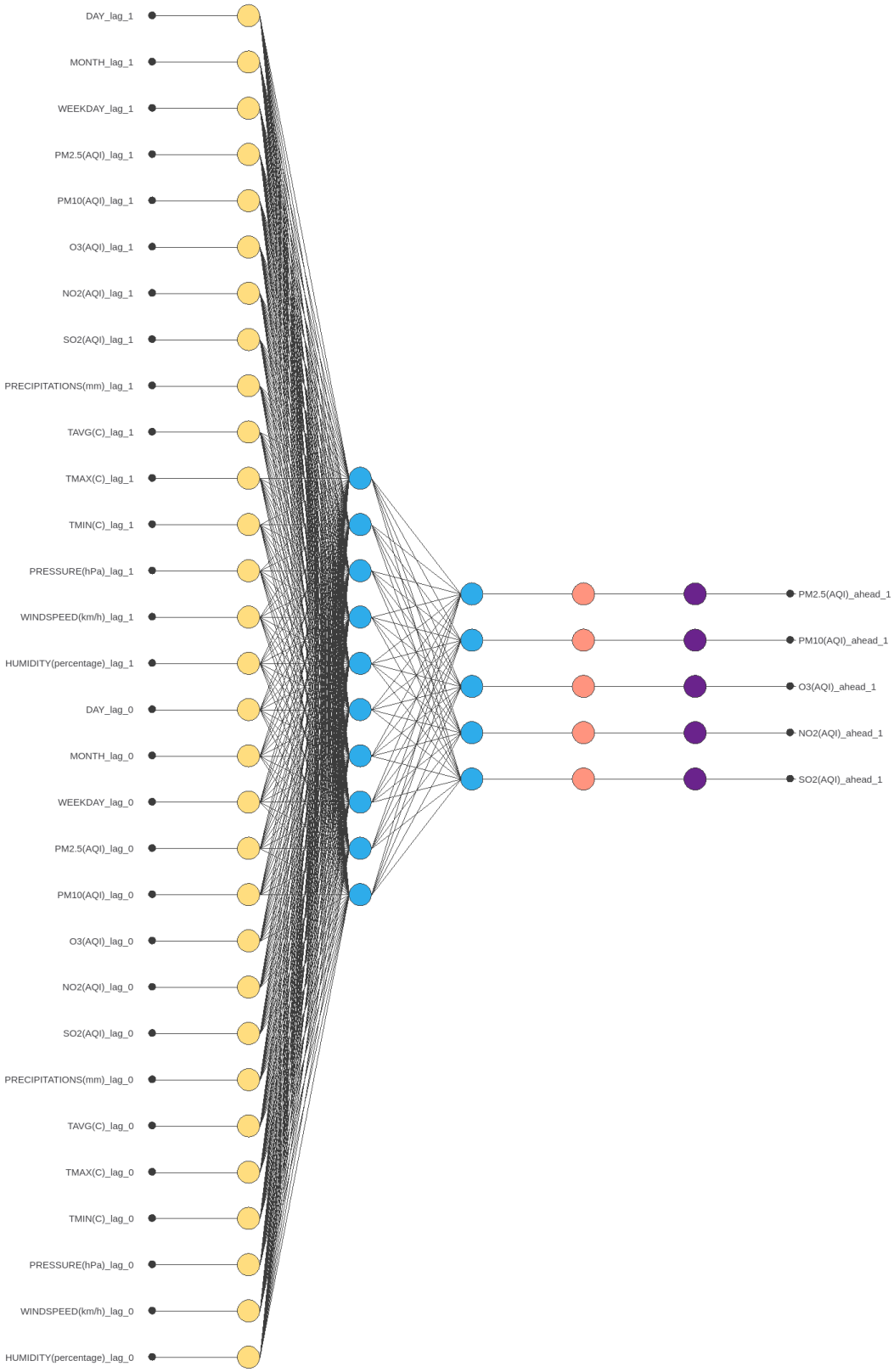
7.2 Datu Basea

Azter dezagun orain erabilitako datu basea. Neural Designer apikazioarekin egindako lanak biltzeko blog bat existitzen da [1], eta blog honeko artikulu zehatz batetik [2] **Ismael Mira**-k egindako lanerako sortua izan zen datu basea erabili da[3]. Honek 2014/01/01-etik 2022/20/07-ra du informazioa (hau da, 2959 informazio lerro) eta informazio honen artean, hurrengo zutabeak aurki ditzakegu:

- DATE: Data formatu egoki batean (Unused).
- DAY: Eguna (Input datua).
- MONTH: Hilabetea (Input datua).
- WEEKDAY: Asteko zenbagarren eguna den (Input datua).
- **PM2.5**(AQI): Kontaminazio parametroa (Input eta **Target** datua).
- **PM10**(AQI): Kontaminazio parametroa (Input eta **Target** datua).
- **O3** (AQI): Kontaminazio parametroa (Input eta **Target** datua).
- **NO2**(AQI): Kontaminazio parametroa (Input eta **Target** datua).
- **SO2**(AQI): Kontaminazio parametroa (Input eta **Target** datua).
- PRECIPITATIONS(mm): Euria (Input datua).
- TAVG(C): Tenperaturaren batazbestekoa (Input datua).
- TMAX(C): Tenperaturaren maximoa (Input datua).
- TMIN(C): Tenperaturaren minimoa (Input datua).
- PRESSURE(hPa): Presioa (Input datua).
- WINDSPEED(km/h): Haizearen abiadura (Input datua).
- HUMIDITY(percentage): Hezetasuna (Input datua).

Zerrenda honetan ikus daitekenez, badaude zenbait zutabe **Target** eta **Input** direna. Demagun modeloak astelehen bateko kontaminazioa aurreikusi nahi duela (Hau da, asteleheneko PM2.5, PM10, O3, NO2 eta SO2 parametroak aurreikusi nahi dituela), horretarako aurreko bi egunetako (larunbatea eta igandea) datuak behar izango dira, kontaminazio parametroak barne. Hortaz, modeloak input bezala 15 datu izan beharren 30 izango ditu, hau

[1] https://www.neuraldesigner.com/blog
[2] https://www.neuraldesigner.com/blog/madrid-air-forecasting
[3] https://www.neuraldesigner.com/files/datasets/madrid-forecasting.csv



7.1 Irudia: Geruza eskalarra 30 neuronekin (horia). Prezeptron geruza 10 neuronekin (urdina). Prezeptron geruza 5 neuronekin (urdina). *Unscaling* geruza 5 neuronekin (gorria). *Bounding* geruza 5 neuronekin (morea).

da, bi egun, eta output bezala 5 datu izango ditu. "DATE"datua Neural Designer aplikazioak behar duelako bakarrik dago datu basean, baino entrenamenduan ez da erabiliko.

Datu base hau osatzeko, 2 api desberdinen laguntza behar izan da:

"https://**api.waqi.info**/feed/here"API-arekin datu baseaz (AQI) ikurraz markatuta dauden parametroak izan dira lortuak, hau da, kontaminazio parametroak, eta aurrerago aurreikusi beharko ditugunak dira hauek.

Bestalde, "http://api.**openweathermap**.org/data/2.5"API-a izan da erabili denaa Madrileko eguraldiari buruzko informazioa lortzeko.

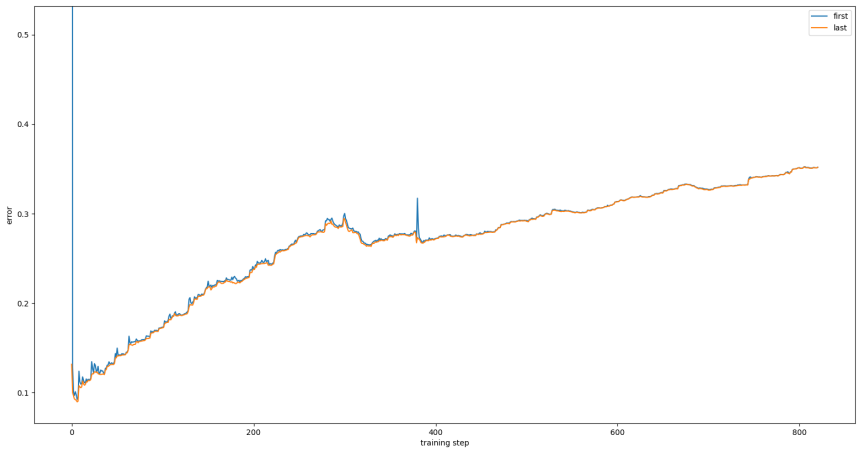
Datu historikoetaz gain, proiektua mikrozerbitzuekin integratu nahi izan ezkero bi API hauek egunez eguneko informazioa eskugragarri izan dute

7.3 Modeloen enrenamendua

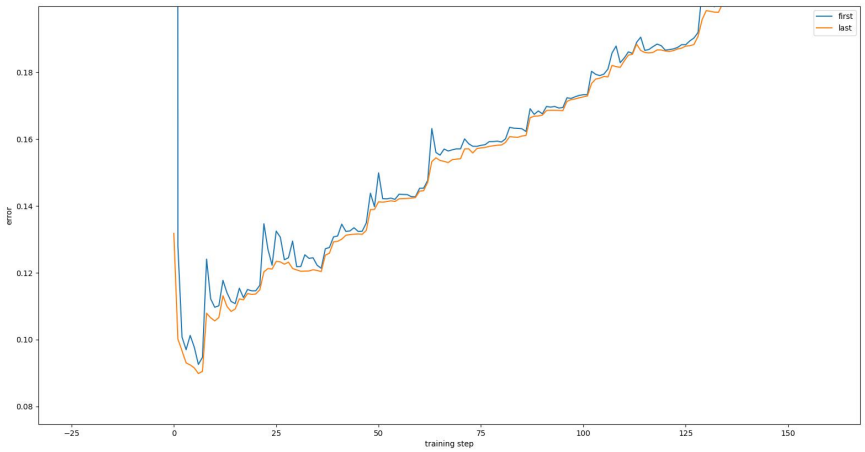
Behin modeloa eta datu basea definiturik daudela, eta lan moduluaren funtzionamendua nolakoa den badakigula (5.1 atalean azaltzen da), kasu errealarekin lanean has gaitezke.

7.3.1 Ikasketa inkrementala

Hasteko ikasketa inkremental normalaren prozesuaz entrenatuko dugu gure modeloa. Horretarako eta aurretik adierazi den bezala, lehen *training step*-ean n daturekin entrenatuko dugu modeloa, ondoren, datu berri bat sartzerakoan (hau da, egun bat pasa dela eta hurrengo eguneko datuak ere ditugula simulatzean) $n + 1$ daturekin entrenatuko dugu datu basea. Prozesu hau jarraitu egingo da $n + 827$ pausura iritsi arte ($n = 353$, hau da, urte natural batetik lortuak izan diren datu posible guztiak jakinik 12 egunetako datuak ezin izan direla eskuratu). 7.2 eta 7.3 irudietan ikus daiteke entrenatu ondoren lorturiko ADAM erroreen grafika.



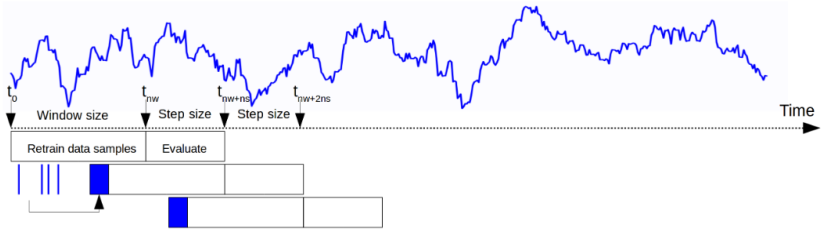
7.2 Irudia: ADAM erroreen grafika. Pausu bakoitzeko, lortutako lehen (urdinez) eta azken (laranjaz) ADAM erroreak grafikatuko dira dira.



7.3 Irudia: ADAM erroreen grafika lehen kurban zoom eginez.

Ikusi daiteke prozesuan zehar lortutako ADAM entrenamendu errorea igoz joan dela, askotan igo eta gero haran bat irudikatuz. Hau patroi berrietara adaptatzearen ondorioa da, izan ere, predikzio arazoei aurre egitea ataza erraza ez da eta ereduak denbora asko edo beste tekniken laguntza beahar izaten dute algoritmoa datu berrietara adaptatzeko.

Lehen emaitza ikusirik, bigarren froga bat egin da, baino kasu honetan Teng, D., & Dasgupta, S. Continuous Time-series Forecasting with Deep and Shallow Stochastic Processes [3] ikerketan jasota dagoen teknika aplikatuz. 7.4 irudian iksuten denez, datu leiho bat finkatzen da, edo beste era batera esanda datu kopuru finko bat, esaterako n, eta une honetatik aurrera modeloak beti n daturekin entrenatuko du.



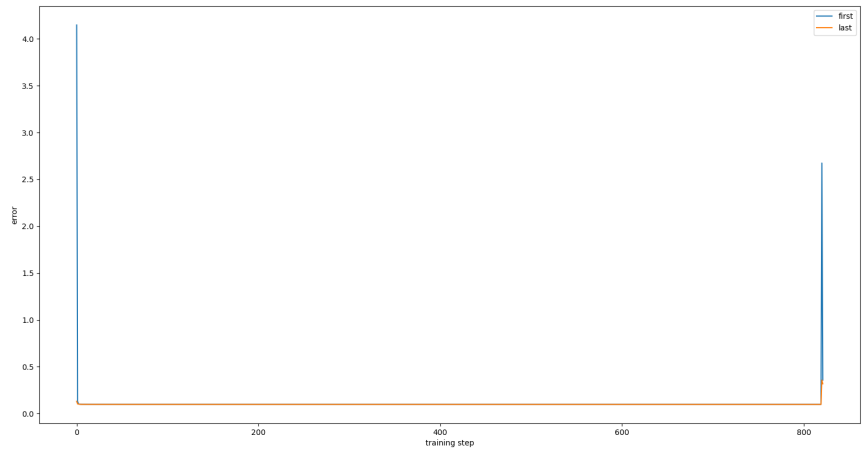
7.4 Irudia: Denbora serie jarraituentzako ikaskuntza inkrementalaren konfigurazioa [3]

Era honetan datu basean aurrera egingo da datu kopuru hau mantenduz eta datuen urtarokotasunera adaptatu egingo da modeloa. Honekin, aurreko aurreko entrenamenduakn gertatu dena ekidin nahi da (zehazki, errorea entrenamenduan zehar igotzea).

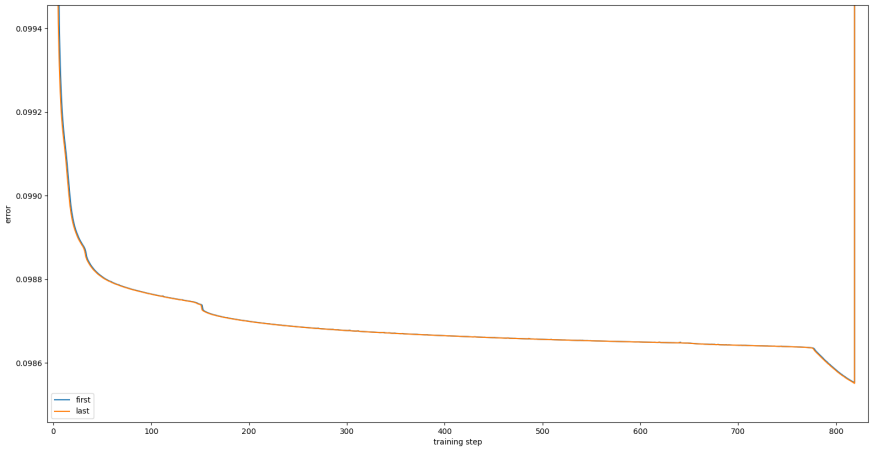
7.3.2 Ikasketa inkrementala & window

Esan den bezala Teng, D., & Dasgupta, S. Continuous Time-series Forecasting with Deep and Shallow Stochastic Processes [3] artikuluan erabilitako teknika aplikatuko da, kasu honetan $n = 354$ datuko leihoa utziz, hau da, datu baseko lehen urtea (365 egunen datuak izatea ez da posible izan zenbait egunetako datuak eskuragarri ez daudelako). Gainera, aurreko ataleko Ikasketa inkrementalari esker lortutako modeloa eta modelo hau geroago

konparatzeko asmoz, $n + 827$ pausura arte iritsiko da entrenamendua, hortaz, 827 egun edo beste era batera esanda 2 urte eta 97 egun pasa direla simulatuko dugu. Entrenamendu hau 7.5 eta 7.6 irudietan ikus daiteke.



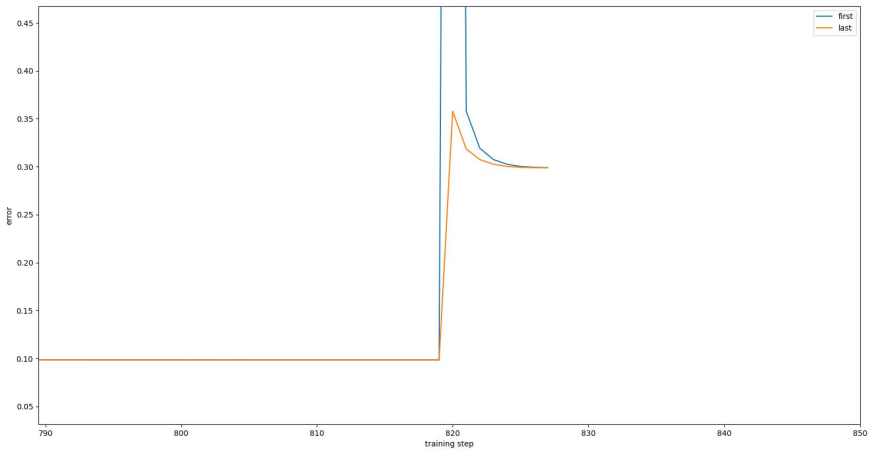
7.5 Irudia: ADAM erroreen grafika. Pausu bakoitzeko, lortutako lehen (urdinez) eta azken (laranjaz) ADAM erroreak grafikatuko dira dira.



7.6 Irudia: ADAM erroreen grafika lehen kurban zoom eginez.

Entrenamendu honen ondoren, hainbat gauza ondoriozta ditzazkegu. Lehenik eta behin ikusi daiteke datu leiho bat entrenamenduari gehituz gero hau asko sendotzen dugula, izan ere, datuek kasu zehatz honetan duten urterokotasunarentzako oso onuragarria da modeloak azken urtaroetako datuekin bakarrik ber-entrenatzea.

Gainera, aurreko kasuarekin aldaratuz, ikus daiteke ere entrenamenduari esker lortutako ADAM erroreen tendentzia beherakorra dela, eta nahiz eta azken hurratsetan anomalia bat suertatu, modeloa ber-entrenatzeko garaian antzeman daiteke segituan modeloak berriz ere egonkortzeko eta errorea jeisteko tendentzia hartu duela 7.7.



7.7 Irudia: ADAM erroreen grafika azken anomalian zoom eginez.

7.1 taulan ikus daiteke honen eragilea izan den datu lerroa.

DATE	DAY	MONTH	WEEKDAY	PM2.5	PM10	O3	NO2	SO2
09/05/2017	9	5	2	53	17	37	19	1

PRECIPITATIONS	TAVG	TMAX	TMIN
0	21.6	29.5	10.7

PRESSURE	WINDSPEED	HUMIDITY
1011.4	5.9	36.8

7.1 Taula: Anomalia sorrarazi duen datu lerroaren edukia

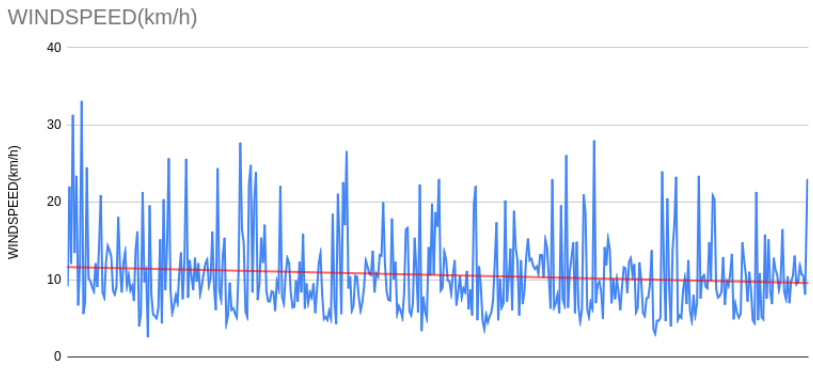
Anomaliaren zergaitia ulertzeko, datu lerroaren datuen artean bariantzak bilatu ditzakegu lerro hau datu base osoarekin konparatuz. Desberdintasun handienen artean hurrengoak aurki ditzakegu:

Lehenik, 7.8 irudian ikus daitekenez, datu baseko beste astearteekin konparatuz haizearen abiadura tendentziaren nahiko azpitik dago. Hau 12 Km/h izanik, kasu honetan haizearen abiadura 6 Km/h-ra ere ez da iristen.

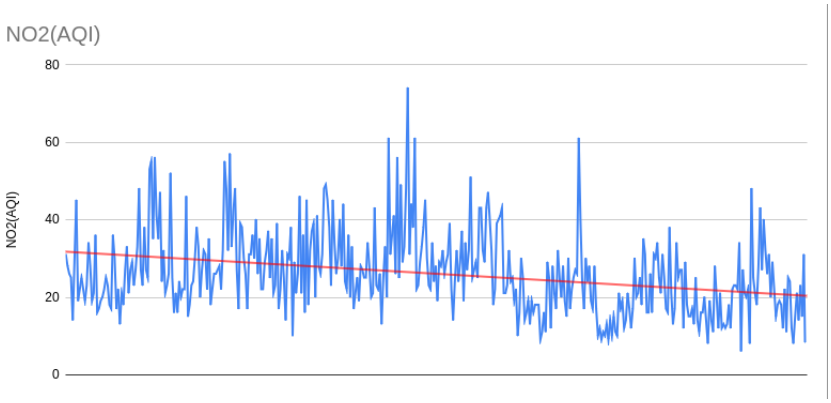
Bestetik, NO2 kontaminazio faktorea dugu. 7.9 irudian ikus daitekenez, honen tendentzia jeitsiz doa urteetan zehar, baino 2017an, 30 ingurukoa zen. Kasu honetan NO2-ak 19 mmarkatzen du, tendentziatik hurrun. Beraz, anomaliaren sortzaile ere izan daiteke NO2-a.

Eta bukatzeko SO2 kontaminazio faktorea dugu. 7.10 irudian ikus daitekenez, 2017 SO2-aren tendentzia 5-ekoa zen, anomaliaren sortzailea izan den datuak 1eko SO2-a izanik.

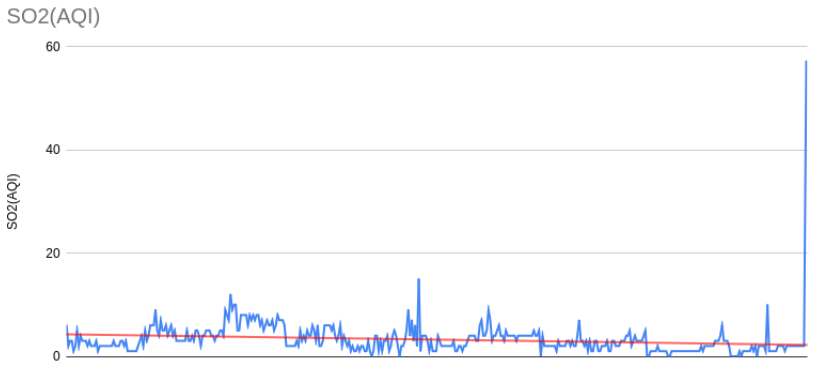
Beraz bai haizearen abiadura, bai NO2-a eta bai SO2-a izan daitezke datu honek anomalia sortzearen errdunak, edo baita ere hauen 3en konbinaketa.



7.8 Irudia: Datu baseko egun guztien haizearen abiadura erakusten duen grafikoa.



7.9 Irudia: Datu baseko egun guztien NO2-a erakusten duen grafikoa.



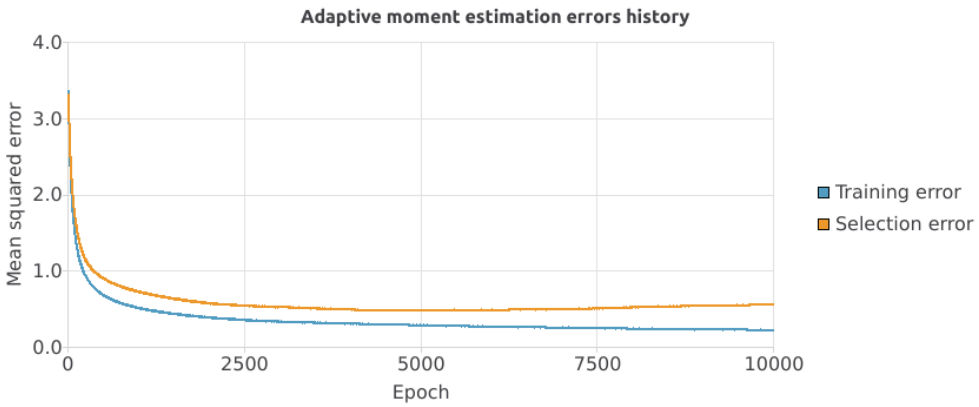
7.10 Irudia: Datu baseko egun guztien SO2-a erakusten duen grafikoa.

7.3.3 Ohiko ikasketa

Behin ikasketa inkrementala erabiliz gure bi modeloak entrenatu ditugula, azken modelo bat entrenatuko dugu, kasu honetan, ohiko ikasketa estandarra erabiliz. Honen zergaitia da bukaeran 3 modelo entrenatu desberdin hauek izaterakoan, hauen predikzioak datu errealekin konparatzea ondorio baliotsuak ateratzeko, eta era berean, ikasketa inkrementalaren aurreikuspenak gutxienez ohiko ikasketaren metodoak bezain ahaltsuak direla frogatzea.

Besterik gabe, esan beharra dago beste bi modeloen antzera hau ere hasieran azaldutako modeloa hartu duela jatorrizko puntu bezala, eta hau entrenatzeko baldintza eta xehetasun berdinak erabili dira.

Logikoa denez, pausu honetan dugun desberdintasun bakarra da entrenamendua behin bakarrrik burutu dela (hasierako n daturekin, hau da, 354 daturekin, eta $n + 827$ datuetara iritsi arte) hortaz, lortutako grafika desberdina da. Kasu honetan entrenamendu bakar bat burutuz joan den bitartean uneoro lortutako errorea grafikatu egingo da.

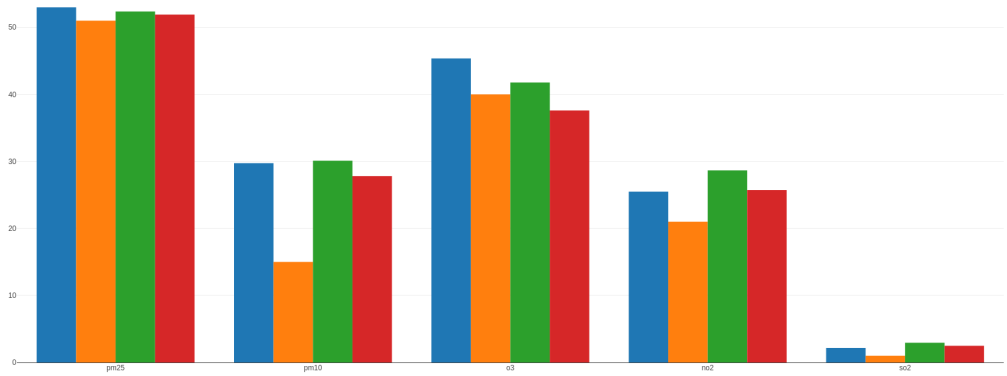


7.11 Irudia: ADAM erroreen historiala urdinez eta *Selection error* neurtzen duen grafika laranjaz (Nahiz eta kasu honetan beharrezkoa ez den Neural Designer aplikazioak grafikoetan gehitzen duen neurketa da) .

7.4 Emaitzen konparaketa

Behin hiru modeloak 2014/01/01-tik 2017/05/16-ra entrenatu ditugula, hauetako bakoitzarekin hurrengo eguneko predikzioa egingo dugu eta datu errealekin konparatuko dugu. Horretarako, *dashboard* bat prestatu da, eta modelo bakoitzaren python errepresentazioaz baliatuz eta aurreko bi egunen informazioa pasaz (2017/05/16 eta 2017/05/15), 2017/05/17-ko Madrilgo kontaminazioa aurreikusten saiatuko da modeloa. Emaitza 7.12 irudian ikus daiteke.

Grafikaren azterketa egiteko, konparaketa taula bat egingo da, emaitza onenak berdez idatziz eta okerrenak berriz gorriz. Hau ulergarriagoa egiteko asmoz, hemendik aurrera Ikasketa inkremental normalaz entrenatutako modeloari I.I.N. deituko diogu, Ikasketa inkrementala eta loeihoaren teknikak nahastuz entrenatutako modeloari I.I.L. deituko diogu eta Ohiko ikasketaz entrenatutako modeloari O.I. deituko diogu.



7.12 Irudia: I.I. normalaren emaitzak (**Urdinez**), Datu errealak (**Laranjaz**), ohiko ikasketaren emaitzak (**Berdez**) eta I.I. leihoaren teknikarekin nahastuz lortutako emaitzak (**Gorriz**) konparatzen dituen grafika.

	PM2.5	**PM10**	**O3**	**NO2**	**SO2**
O.I.	52.37	**30.1**	**41.8**	**28.7**	**2.94**
I.I.N.	**52.99**	29.74	**45.36**	**25.49**	**2.16**
I.I.L.	**51.9**	**27.8**	37.6	25.72	2.48
Datu errealak	51.0	15.0	40.0	21.0	1.0

7.2 Taula: I.I.N, I.I.L. eta O.I. modeloen emaitzen konparaketa

Nahiz eta 3 modeloen emaitzak datu errealekin konparatuz nahiko gertutasuna izan duten (salbuespenak salbuespen), azter ditzagun 3 hauek sakonago:

- Hasteko, O.I. modeloak aurreikusi beharreko bost kontaminazio parametro posibleetatik, O3-arekin bakarrik izan da emaitza gertuena lortu duena. Bestalde, ikus dezakegu ere bai PM10-arekin, bai NO2-arekin eta bai SO2-arekin emaitza txarrenak lortu dituen modeloa izan dela.

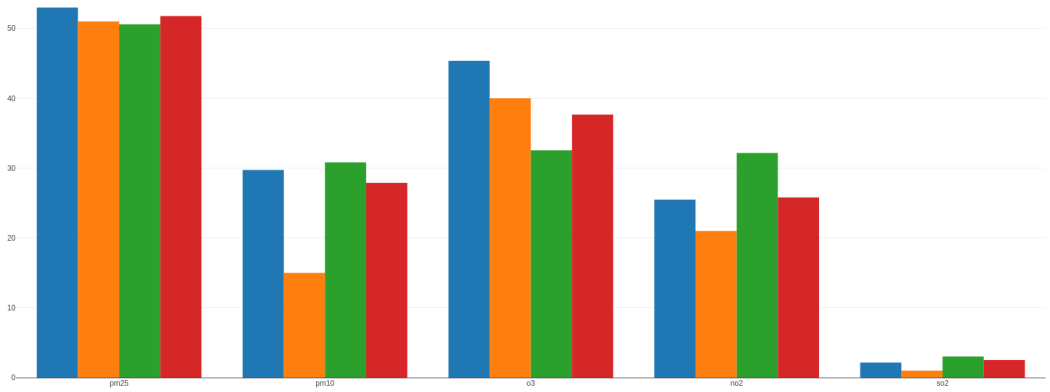
- Bestetik I.I.N. modeloa dugu. Honek, bi parametrorekin izan du hiru modeloetatik emaitzarik onena, NO2-arekin eta SO2-arekin, eta beste bi parametrorekin izan du emaitzarik txarrena (beste modeloekin aldaratuz) PM2.5 parametroarekin, eta O3 parametroarekin.

- Azkenik I.I.L. modeloa dugu. Modelo honek aurrekoaren antzera bi paramterorekin izan du emaitzarik onena (PM2.5 eta PMM10) baino behin ere ez du beste modeloekin aldaratuz emaitza txarrena izan.

Behin konporaketa hau aztertu egin dugula, bigarren esperimentu bat egingo dugu *dashboard* berdina berrerabiliz eta aurreko pausuan azaldutako modeloen python errepresentazioetaz baliatuz.

Kasu honetan, 3 modeloei egun bat gehiago gehituko zaie datu basean eta dagoeneko aztertu diren xehetasunak errespetatuz, berriro entrenatuak izango dira. Orain modeloek 2017/05/17-ra arteko informazioaz baliatuz izan dira entrenatuak, eta azken bi egunak

kontuan izanik, hau da, 2017/05/17 eta 2017/05/16, 2017/05/18-ko Madrilgo kontaminazioa aurreikusten saiatuko da modeloa. Emaitzak 7.13 irudian ikus daitezke.



7.13 Irudia: I.I. normalaren emaitzak (**Urdinez**), Datu errealak (**Laranjaz**), ohiko ikasketaren emaitzak (**Berdez**) eta I.I. leihoaren teknikarekin nahastuz lortutako emaitzak (**Gorriz**) konparatzen dituen grafika.

	PM2.5	PM10	O3	NO2	SO2
O.I.	**50.59**	**30.82**	**32.55**	**32.17**	**3.02**
I.I.N.	**52.99**	29.73	45.36	**25.49**	**2.16**
I.I.L.	51.77	**27.89**	**37.66**	25.80	2.53
Datu errealak	51.0	15.0	40.0	21.0	1.0

7.3 Taula: I.I.N, I.I.L. eta O.I. modeloen emaitzen konparaketa

Nahiz eta 3 modeloen emaitzak datu errealekin konparatuz nahiko gertutasuna izan duten (salbuespenak salbuespen), azter ditzagun 3 hauek sakonago:

- Hasteko, O.I. modeloak aurreikusi beharreko bost kontaminazio parametro posibleetatik, PM2.5-arekin bakarrik izan da emaitza gertuena lortu duena. Bestetik, ikus daiteke beste kontaminazio parametro guztiak aztertuz, beste modeloekin aldaratuz emaitza txarrenak lortu duena izan dela.
- Hurrengoa I.I.N. modeloa da. Honek, aurreko azterketa kasuan bezala, bi parametrorekin izan du hiru modeloetatik emaitzarik onena, NO2-arekin eta SO2-arekin. Bestetik, bi parametrorekin izan du emaitzarik txarrena beste modeloekin aldaratuz, PM2.5 parametroarekin eta O3 parametroarekin.
- Azkenik I.I.L. modeloa aztertuko dugu. Ez dago kontaminazio parametrorik zeinarentzako modelo honek predikzio txarrena egin duenik, aldez, bai PM10-arentzako eta O3-arentzako predikzio gertuena egin du.

Dagoeneko beharrezko esperimentuak eginak daude, eta hauek aztertuz, hurrengoa aztertu dezakegu:

Madrilgo kontaminazioa ondorioztatzearen kasu zehatz honetarako, esan daiteke bai lehen esperimentuan eta bai bigarren esperimentuan gutxien failatu duen modeloa Ikasketa

Inkrementala erabiliz eta leihoaren teknika nahasiz entrenatua izan den modeloa izan dela. Aldiz, ondoriozta daiteke ere gehien failatu duen modeloa oinarrizko ikasketa burutuz entrenatu dena izan dela. Beraz, esan dezakegu kasu zehatz onetarako Ikasketa inkrementala gutxienez oinarrizko ikasketa bezain ona dela.

KAPITULUA 8

Ondorioak eta etorkizuneko aplikazio posibleak

Lan honetan zehar MLOps-aren kontzeptua ikertu egin da lehenik ikasketa inkrementalaren bi kasu teoriko landuz (sinu funtzioaren eta funtzio exponentzialaren aproximazioa) eta ondoren Madrilgo kontaminazioa aurreikusten saiatuz modelo bat ikasketa inkrementalaren bidez entrenatuz ere. Honetaz gain, entengabeko monitorizazioa bezalako MLOps teknikarentzako hain garrantzitsua den atala landua izan da, lan moduluak hau nola egiten duen definiturik. Argi utzi beharra dago, ikasketa inkrementalaren muina, modeloak etengabe ikasi dezatela delaa, aurreikusten, aproximatzen edota klasifikatzen dauden modeloak sarrerako datuetara adaptatzen diren bitartean, hortaz, proiektu honen bidez hori da erakutsi nahi izan dena, aurreikusketa modelo batek madrilgo kontaminazioa nola aurreikusten duen ikusiz urtaro bakoitzeko datuetara adaptatuz leihoaren teknika erabiliz.

Gainera, teknika honek MLOps paradigmarekin duen erlazio estua erakutsia izan da ere, ikasketa inkrementaleko modeloa paradigma honetan integratuz eta mikrozerbitzuen erabilpena eta erabilgarritasuna MLOps-arentzako erakutsiz. Honetaz gain, MLOps bidez modeloen garapena produzkio inguruneetarana nola adaptatu daitezkeen erakutsi da *continious deployment*-aren kontzeptua landuz besteak beste.

Azkenik, lanean zehar ikusi denez, gaur egungo gizarteak geroz eta datu gehiago sortzen ditu, hauen prozesamendua ezinbestekoa izanik informatikan. Aurreikusketa modeloek, geroz eta gehiago eguneratuz joan beharko dira eta horretarako datu ugari hauen beharra izango dute. Honengatik ikasketa inkrementala hala nola MLOps teknikak informatikan geroz eta gehiago ikertuko diren atalak izango dira lan honetan egin den ikerketa eta inplementazio lana geroz eta garrantzia gehiago hartuz informatikak izango dituen beharretan.

Lan honen hurrengo garapen pausuak etorkizunera begira ugariak dira, esaterako, Neural desginer aplikazioan integratu nahi da lan modulu hau. Era honetan, madrilgo kontaminazioarekin egin den azterketa, Neural designer aplikazioan sortzen den edozein proiektuarekin egin ahalko litzateke. Lortutako emaitzak baliagarriak eta oso interesgarriak dira, beraz, lan honi esker, ikasketa inkrementala MLOps paradigmaren barruan Neural Designer aplikazioan integratzeko aurrera pauso bat suposatzea espero da.

Bibliografia

[1] Rudy Semola, Vincenzo Lomonaco, and Davide Bacciu. Continual-learning-as-a-service (claas): On-demand efficient adaptation of predictive models. *arXiv preprint arXiv:2206.06957*, 2022. Ikusi v, 16, and 17 orrialdeak.

[2] Yizheng Huang, Huaizheng Zhang, Yonggang Wen, Peng Sun, and Nguyen Binh Duong TA. Modelci-e: Enabling continual learning in deep learning serving systems. 2021. Ikusi v, 17, and 18 orrialdeak.

[3] Dan Teng and Sakyasingha Dasgupta. Continuous time-series forecasting with deep and shallow stochastic processes. Ikusi v, 39 orrialdeak.

[4] Jeffrey Schlimmer and Doug Fisher. A case study of incremental concept induction. pages 496–501, 01 1986. Ikusi 1 orrialdea.

[5] Dennis Lock. *Project management*. Routledge, 2020. Ikusi 5 orrialdea.

[6] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *IEEE Software*, 33(3):94–100, 2016. Ikusi 13 orrialdea.

[7] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 11:31866–31879, 2023. Ikusi 14 orrialdea.

[8] Sridhar Alla and Suman Kalyan Adari. *What Is MLOps?*, pages 79–124. Apress, Berkeley, CA, 2021. Ikusi 14 orrialdea.

[9] Yong Luo, Liancheng Yin, Wenchao Bai, and Keming Mao. An appraisal of incremental learning methods. *Entropy*, 22(11), 2020. Ikusi 15 orrialdea.

[10] Shubhanshu Mishra and Jana Diesner. Pytail: Interactive and incremental learning of nlp models with human in the loop for online data, 2022. Ikusi 16 orrialdea.

[11] Eulogio Gutierrez-Huampo, Danilo S. Barbosa, and Orlando Chuquimia. Real-time class-incremental learning for voice command recognition via adaptive oisgng. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. Ikusi 16 orrialdea.

[12] Charles Anderson. Docker [software engineering]. *IEEE Software*, 32(3):102–c3, 2015. Ikusi 20 orrialdea.

[13] Joshua Cook. *The Dockerfile*, pages 81–101. Apress, Berkeley, CA, 2017. Ikusi 25 orrialdea.

[14] Shimon Ifrah. *Deploy Containers on AWS*. Apress, Berkeley, CA, 1 edition, 2019. Ikusi 25 orrialdea.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2017. Ikusi 35 orrialdea.