Vloga MLOps pri zagotavljanju kakovosti napovednih modelov

Zala Lahovnik, Grega Vrbančič, Vili Podgorelec

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko,
Maribor, Slovenija
zala.lahovnik1@um.si, grega.vrbancic@um.si, vili.podgorelec@um.si

Z razvojem umetne inteligence in strojnega učenja ter vse pogostejšo integracijo v programske rešitve, postaja eden izmed ključnih vidikov zagotavljanje zanesljivosti in kakovosti napovednih modelov strojnega učenja. Kljub napredku se številni projekti, ki vključujejo inteligentne komponente soočajo z neuspehi zaradi pomanjkljivega testiranja in neustrezne integracije v produkcijska okolja. Pogosti razlogi za neuspeh vključujejo slabo integracijo modelov v informacijske sisteme, pomanjkljivo testiranje in odsotnost sistematičnega nadzora kakovosti modelov. Ključno vlogo pri uspešnosti modelov ima kakovost vhodnih podatkov – napačni, nepopolni ali pristranski podatki zmanjšajo robustnost napovedi in posledično zaupanje uporabnikov. Pomemben izziv predstavlja tudi pojav spremembe porazdelitev vrednosti podatkov, ki lahko vodi v postopno degradacijo modela. Te izzive naslavlja MLOps pristop, ki združuje pristop DevOps z značilnostmi razvoja in upravljanja napovednih modelov strojnega učenja. MLOps omogoča avtomatizacijo, ponovljivost, transparentnost in nadzor celotnega življenjskega cikla napovednih modelov. Ključni gradniki vključujejo verzioniranje podatkov in modelov, validacijo, avtomatizirano ponovno učenje ter sprotno spremljanje uspešnosti delovanja v produkciji. V prispevku so ti vidiki osvetljeni skozi študijo primera Airly platforme, ki predstavlja celovit primer MLOps podpore.

Ključne besede:
MLOps,
zagotavljanje kakovosti,
napovedni modeli,
strojno učenje,
inteligentni sistemi.

Univerzitetna založba Univerze v Mariboru

ISBN 978-961-299-037-4
DOI https://doi.org/10.18690/um.feri.7.2025.1

1 Uvod

V zadnjem desetletju je razvoj umetne inteligence (angl. Artificial Intelligence, AI) in strojnega učenja (angl. Machine Learning, ML) povzročil pomembne premike na številnih področjih, kot so medicina, industrija, finance in transport. Kljub temu pa številni AI/ML projekti v praksi ne dosegajo pričakovanih rezultatov in pogosto ne uspejo preiti iz raziskovalne faze v produkcijska okolja. Raziskave kažejo, da do 85 % projektov na področju umetne inteligence ne uspe doseči trajne implementacije ali želenih poslovnih učinkov. Glavni vzroki neuspeha so pogosto povezani z neustrezno integracijo modelov v obstoječe informacijske sisteme, pomanjkljivim testiranjem in odsotnostjo sistematičnega nadzora kakovosti modelov v realnem času [1].

Posebna značilnost inteligentnih sistemov je izrazit vpliv kakovosti vhodnih podatkov na učenje in posledično tudi na uspešnost napovedovanja modelov. Napačni, nepopolni ali pristranski podatki lahko pomembno poslabšajo uspešnost in robustnost napovednih modelov, kar vodi v nezanesljive rezultate in zmanjšano zaupanje uporabnikov [2]. Pogosto se modeli soočajo s tudi pojavom spremembe porazdelitve vrednosti podatkov (angl. data drift). To je pojav pri katerem se s časom spreminjajo porazdelitve vrednosti vhodnih podatkov ali razmerja med vhodnimi in izhodnimi značilnicami. Brez ustreznega nadzora in prilagajanja lahko takšne spremembe povzročijo postopno degradacijo napovedne uspešnosti modela, kar lahko pripelje do napačnih odločitev ter posledično do finančnih izgub [3].

Kot odziv na te izzive se razvija praksa MLOps, ki združuje metodologije DevOps z značilnostmi razvoja in vzdrževanja ML modelov, s ciljem povečanja stopnje avtomatizacije, nadzora kakovosti in zanesljivosti napovednih modelov. MLOps omogoča avtomatizacijo in standardizacijo procesov ter zagotavlja transparentnost, ponovljivost in zanesljivost celotnega življenjskega cikla ML sistemov [4]. Ključne komponente MLOps vključujejo upravljanje verzij modelov in podatkov, avtomatizirane postopke za ponovno učenje modelov, validacijo in testiranje podatkov ter neprekinjeno dostavo in spremljanje delovanja modelov kot tudi kakovosti vhodnih podatkov v produkcijskih okoljih.

V prispevku podrobneje obravnavamo ključne izzive in rešitve za zagotavljanje kakovosti napovednih modelov z vidika MLOps praks. Posebno pozornost namenjamo predvsem validaciji, testiranju in nadzoru podatkov, verzioniranju modelov in podatkov ter tehnikam spremljanja modelov v produkcijskih okoljih. Za ilustracijo praktične uporabe predstavljamo študijo primera platforme Airly, ki vključuje celovito MLOps podporo za vse ključne korake v ciklu neprekinjenega učenja napovednih modelov in zagotavljanja njihove zanesljivosti v realnem času.

2 Neprekinjeno učenje

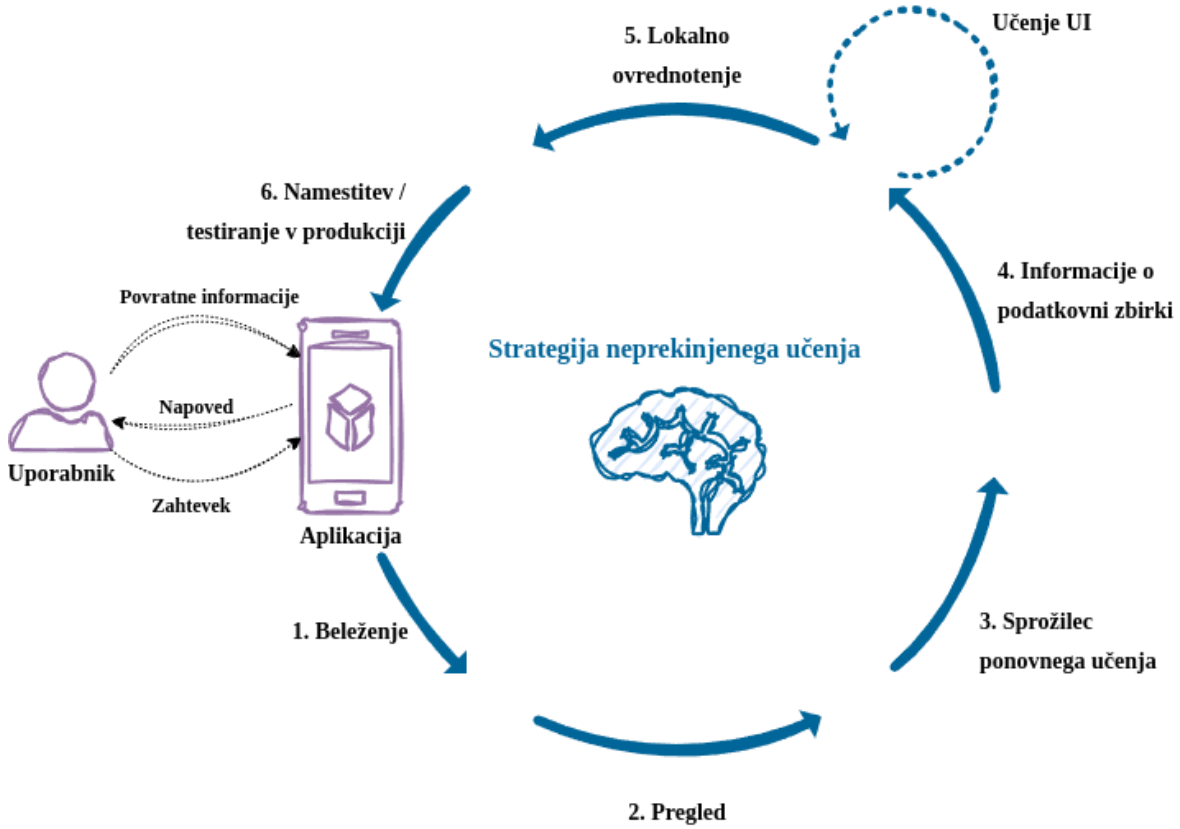
V klasičnem programskem inženirstvu končni izdelek običajno predstavlja statično različico programske rešitve. V nasprotju s tem pa so napovedni modeli strojnega učenja dinamični, njihova napovedna uspešnost pa se pogosto degradira skozi čas zaradi sprememb porazdelitev vrednosti v podatkih, poslovnih pravilih, sprememb v vedenju uporabnikov ali sprememb v okolju. Takšne spremembe zahtevajo neprekinjen nadzor podatkov, posodabljanje napovednih modelov, v mnogih primerih pa tudi ponovno učenje ali do-učenje modelov. Neprekinjeno učenje (angl. Continuous Learning) v kontekstu produkcijskih ML sistemov pomeni vzpostavitev avtomatiziranega sistema, ki vključuje zajem in verzioniranje podatkov, validiranje in testiranje podatkov, ponovljivost in sledenje eksperimentom, ovrednotenje in verzioniranje napovednih modelov, namestitev v produkcijo ter nadzorovanje delovanja modelov v realnem času.

Celoten cikel mora biti ponovljiv, sledljiv in kar se da avtomatiziran, kar predstavlja temelj sodobnih MLOps praks. Takšen pristop omogoča, da modeli ostanejo posodobljeni, prilagojeni trenutnim razmeram in kar se da zanesljivi za uporabo v realnih okoljih. Slika 1 prikazuje ključne korake strategije neprekinjenega učenja, ki so temelj sodobnih

MLOps pristopov. Namen takšne strategije je zagotoviti, da napovedni modeli ohranjajo svojo kakovost tudi po implementaciji v produkcijsko okolje, kjer se podatki in ostali dejavniki pogosto spreminjajo.

Proces se začne z beleženjem podatkov o interakciji uporabnikov z aplikacijo (točka 1), kjer se zbirajo vhodni podatki, napovedi modela in povratne informacije. Ti podatki se nato periodično pregledajo (točka 2) z namenom preverjanja sprememb v porazdelitvi vrednosti podatkov ali sami strukturi podatkov.

Ko sistem zazna pomembne spremembe ali padec napovedne uspešnosti modela, se sproži proces (ponovnega) učenja (točka 3). To vključuje pripravo in verzioniranje nove (posodobljene) verzije podatkovne zbirke (točka 4), kar je ključno za zagotavljanje ponovljivosti eksperimentov in sledljivosti spremembam. Zatem sledi lokalno ovrednotenje posodobljenega napovednega modela (točka 5), pri čemer se nova različica modela primerja z obstoječo glede na izbrane metrike napovedne uspešnosti. Če novi model dosega boljše rezultate, se namesti oziroma posodobi v produkcijskem okolju (točka 6). V okviru MLOps se pri tem uporabi infrastruktura za avtomatizirano testiranje, sledenje modelov in nadzorovanje njihovega vedenja v realnem času. Sistem lahko tako neprekinjeno spremlja napovedno uspešnost modela in pravočasno zazna morebitno degradacijo, kar omogoča proaktivno vzdrževanje kakovosti napovedne uspešnosti ML modelov.



Slika 1: Primer strategije neprekinjenega učenja napovednega modela.

2.1. MLOps

MLOps je skovanka za ML operacije (Ops) in razvoj (Dev), kjer se aplicirajo DevOps prakse na ML metode. DevOps je skupek metod, ki zaznavajo spremembe v programski opremi in upravljajo zanesljivost drugih orodij, ki so s tem povezane. Neprekinjena integracija in neprekinjena dostava uporablja vrsto strategij za zagotovitev, da so izdaje programske opreme hitrejše in zanesljivejše. MLOps metoda vključuje v proces programskega razvoja tudi ML modele. MLOps proces integrira ML značilnosti z DevOps koncepti, s čimer se omogoči avtomatska namestitev in učinkovito nadzorovanje ML modelov v času razvoja. MLOps sisteme sestavljajo 3 glavne komponente, in sicer: model, koda in podatki. Proces MLOps pa nam omogoča tudi enostavnejše nadzorovanje in upravljanje večjega števila napovednih modelov [5].

Ključen dejavnik, ki ima velik vpliv na napovedno uspešnost ML modela, so podatki. Zaradi napak ali anomalij, ki se lahko pojavijo v podatkih, se morajo ti filtrirati in procesirati preden so uporabljeni za učenje napovednih modelov. V samem procesu učenja napovednega modela so vključeni tudi konvencionalni integracijski testi in testi enot. Validacija in analiza ovrednotenja napovedne uspešnosti modela je nujno potrebna za preverjanje delovanja ključne funkcije modela - napovedovanja. Ko model prestane faze ovrednotenja napovedne uspešnosti ter vse integracijske teste in teste enot, se ga namesti v testno okolje, ki simulira delovanje produkcijskega okolja [5]. Grajenje ali učenje napovednega modela zahteva več komponent, med ključne pa spadajo: podatkovna zbirka namenjena za učenje, izbrane metrike napovedne uspešnosti, izbran algoritem strojnega učenja, vrednosti hiperparametrov algoritma strojnega učenja in podatkovna zbirka namenjena za ovrednotenje napovedne uspešnosti (testiranje) naučenega modela. Že samo število komponent, ki sestavlja proces gradnje, je pomemben razlog zakaj je vpeljava MLOps procesa velik izziv. Kompleksnost posameznih komponent pa proces samo še oteži [6].

Modeli, ne glede na uporabljen algoritem, so lahko samo le tako »dobri« kot so »dobri« učni podatki. Različni dejavniki lahko povzročijo, da so podatki neuporabni za namen učenja napovednih modelov, med ključne pa spadajo nepopolnost, nenatančnost in nekonsistentnost podatkov. V ta namen se podatki v procesu MLOps validirajo, testirajo in predprocesirajo. Omenjeni koraki zajemajo pregled distribucije vrednosti podatkov, čiščenje, zapolnjevanje manjkajočih vrednosti, transformacija v drugačno obliko, filtriranje, vzorčenje, itd. [6].

Modeli morajo biti tudi testirani oz. ovrednoteni. To je proces, v katerem preverjamo tudi, ali model sploh dela to, kar bi naj in kako dobro to delo opravlja. Ovrednotiti ga je potrebno v kontekstu njegovega delovanja, pri čemer moramo imeti možnost primerjave z nečim, kar je obstajalo pred modelom – ali prejšnja verzija modela ali rešitev na podlagi pravil, da lahko preverimo kaj bi bil rezultat, če bi nov model zamenjal prejšnjo rešitev. Pri tem je pomembna tudi izbira primerne metrike s katero bomo ocenili in primerjali različne modele glede na dan problem, ki ga rešujejo. Da preverimo sposobnost generalizacije (sposobnosti posplošitve znanja) modela, mora biti metrika izračunana na podlagi podatkovne zbirke, ki ni bila uporabljena v procesu učenja napovednega modela [6].

2.2. Verzioniranje in ponovljivost eksperimentov

Verzioniranje in ponovljivost eksperimentov naslavljata dve različni potrebi in sicer potrebo, da imajo podatkovni znanstveniki možnost vračanja na različne verzije modelov ter da bodo v neki točki razvoja, podatkovni znanstveniki skoraj zagotovo morali poustvariti pogoje, ki so vodili do posodobljene različice napovednega modela, tudi nekaj let po dejanski namestitvi. Predvsem slednja služi za namen primerjave novo razvitih napovednih modelov s predhodnimi.

Verzioniranju je sicer do določene mere že zadoščeno, v kolikor vse temelji na programski kodi, pri čemer so v uporabi orodja za verzioniranje programske kode. Seveda, pa je tipično za namen uspešne poustvaritve pogojev potrebno poskrbeti tudi za verzioniranje podatkovnih zbirk, ki so v nekem trenutku bile uporabljene za učenje napovednega modela. Takšno poustvaritev pogojev z drugimi besedami imenujemo tudi ponovljivost eksperimentov in je ena izmed ključnih lastnosti MLOps paradigme. Torej v vsaki točki v razvoju, ko podatkovni znanstveniki ustvarijo (naučijo) model, ki uspešno rešuje problem, je potrebno zagotoviti, da je model mogoče poustvariti. V namen zagotovitve je potrebno zadostiti številnim vidikom. Prvi je zagotovitev, da so predpostavke podatkovnih znanstvenikov zabeležene. Na ta način poskrbimo, da lahko drug podatkovni znanstvenik brez težav poustvari katerikoli eksperiment, brez dodatnih informacij. Drug vidik je vpliv naključnosti. Mnogi algoritmi strojnega učenja kot tudi sami procesi znotraj razvoja, kot je na primer naključna razdelitev podatkovnih zbirk, uporabljajo psevdo-naključna števila. Da lahko natančno reproduciramo eksperiment, moramo imeti kontrolo tudi nad temi števili. To največkrat kontroliramo z nastavljanjem fiksne vrednosti semena (angl. seed) v generatorju števil. S stališča podatkov, kot omenjeno potrebujemo podatke v obliki kot so bili v tistem času. Prav tako ne smemo zanemariti raznih nastavitev metod ali algoritmov, ki so uporabljeni v celotnem procesu. Tudi te je potrebno skrbno shranjevati saj lahko zgolj na tak način zagotovimo ponovljivost eksperimentov. Za namen

primerjave napovednih uspešnosti posameznih verzij napovednih modelov je potrebno beleženje oz. shranjevanje tudi teh. Nepresenetljivo tudi različne implementacije enakega modela vodijo v različne modele, ki lahko nato v določenih primerih ob istih vhodnih podatkih vrnejo napovedi, ki so si med seboj različne. In nenazadnje je potrebno tudi zagotoviti enako izvajalno okolje. Že nekoliko različne verzije Python knjižnic lahko vodijo v drugačne rezultate, kar pa je težko napovedati in nadzorovati [6].

Verzioniranje modelov je pomemben vidik MLOps, ki omogoča, da modelom sistematično sledimo in jih nadziramo. Uporaba verzioniranja modelov je nujna za zagotavljanje ponovljivosti, nadzorovanja posodobitev ali sprememb in izvedbo vrnitve na predhodno verzijo (angl. rollback). Poznamo različne strategije poimenovanja verzij modela, kot je semantično verzioniranje, ki uporablja shemo za številko verzije (npr. major.minor.patch) ali pa se uporablja strategija z uporabo metapodatkov, ki vključujejo različne podrobnosti o modelu, kot je arhitektura, učni podatki in hiperparametri. Slednji pristop, ki je tudi med bolj uporabljanimi, v ta namen uporablja t.i. register modelov. Register modelov služi kot odložišče vseh prej omenjenih informacij oz. podatkov, ki so potrebni za uspešno reproduciranje nekega eksperimenta [7]. Ena izmed bolj popularnih odprtokodnih rešitev je kot je MLFlow, ki ponuja centralizirano orodje za shranjevanje, upravljanje in dostop do različic modelov, obstaja pa tudi množica lastniških in plačljivih rešitev kot so Neptune, Metaflow, ipd..

Verzioniranje podatkov sledi enaki paradigmi kot verzioniranje programske kode, vendar za podatkovne zbirke. Sistemi z živimi podatki konstantno pridobivajo nove podatke, kar lahko vodi v različne verzije enake podatkovne zbirke. Verzioniranje podatkov omogoča sledenje podatkovnim zbirkam, kjer se zabeleži vsaka sprememba na posamezni podatkovni zbirki. S tem procesom pridobimo vpogled v spremembe in razvoj podatkov v projektu in zmanjšamo tveganja za nastanek napak. Če se pojavi napaka z najnovejšo verzijo podatkov, se lahko enostavno vrnemo na zadnjo delujočo verzijo. Z uporabo tehnologij verzioniranja podatkov lahko ekipe zajamejo verzije podatkov in modelov z Git commit-i, kar predstavlja mehanizem za spremembo med različnimi podatki. Rezultat takšnih tehnologij je ena zgodovina tako za podatke, kot za kodo in za modele strojnega učenja, skozi katere se lahko pomikajo člani ekipe [8]. Med pogosteje uporabljanimi orodji za namen verzioniranja podatkovnih zbirk so Git-LFS in DVC, ki se oba relativno enostavno integrirata v orodje Git.

2.3. Učenje napovednih modelov

Uporaba napovednih modelov strojnega učenja je iz dneva v dan bolj vseprisotna. Specifična lastnost napovednih modelov pa je, da tipično niso statični in jih je tega potrebno pogosto ponovno učiti, da se prilagodijo na spremembe v podatkih [9]. Ponovno učenje modela je nujno, ko se pojavi sprememba v porazdelitvi vrednosti podatkov (angl. Data Drift). Ta pojav se lahko pripeti že zaradi sezonskosti podatkov ali nenazadnje sprememb v uporabniških nastavitvah. Ko pride do takšnega pojava se tipično napovedna uspešnost modela zmanjša [11].

Ponovno učenje modelov je pomembna komponenta za prilagajanje modelov. Modele lahko ponovno učimo po vnaprej določenem časovnem obdobju na enake intervale ne glede na napovedno uspešnost modelov. Takšen pristop je sicer predvidljiv (načrtovan), vendar lahko vodi v zakasnjeno prilagoditev napovednega modela na spremembe v okolju (podatkih). Načrtovano ponovno učenje se dogaja v določenih intervalih, kot je tedensko, mesečno, glede na zgodovinske trende v podatkovnih premikih [10]. Takšen pristop je uporaben, kadar je podatkovni premik možno napovedati in se zgodi postopoma. Pri takšnem podatkovnem premiku lahko pripravimo sistematične posodobitve, ki ne bodo močno vplivale na produkcijski cevovod. Omejitev omenjenega pristopa pa je, da se ne more dinamično odzivati na spremembe v podatkih. Če se podatkovni premik zgodi prej kot pričakovano bo model slabše deloval do naslednje načrtovane posodobitve. Podobna omejitev se zgodi tudi, če je interval prevelik. V takšnem primeru model morda ne bo uspel zajeti pomembnih sprememb v podatkih, zaradi česar bo dalj časa deloval s slabšo točnostjo. Zaradi takšnih pomanjkljivosti omenjenega pristopa so bile predlagane metode, kjer se modeli dinamično ponovno učijo glede na določene prage napovedne uspešnosti (angl. performance thresholds) ali zaznanih sprememb v porazdelitvah vrednosti podatkovnih. Slednji pristop pa seveda zahteva, neprekinjeno nadzorovanje in ovrednotenje napovednih modelov v produkciji (angl. online evaluation),

saj sicer nimamo vzvoda, na podlagi katerega, bi lahko objektivno ovrednotili uspešnost delovanja napovednega modela [12].

2.4. Namestitev v produkcijo

Ko je model ponovno naučen, sledi namestitev v izvajalno okolje. Preden ga namestimo je potrebno preveriti, da model ne vpeljuje nazadovanja v smiselne napovedne uspešnost ali da ne vpeljuje pristranskosti. S tem namenom izvajamo ovrednotenje modela pred dejansko izpostavitvijo izbranega modela vsem končnim uporabnikom. V ta namen lahko uporabimo obstoječe pristope testiranja, kot na primer AB testiranje kjer nov model izpostavimo manjši podmnožici uporabnikov, ostali pa še vedno uporabljajo trenutno verzijo napovednega modela. S tem zmanjšamo tveganje za nastanek poslovne škode v primeru slabšega delovanja posodobljenega napovednega modela, saj mu je izpostavljen le manjši delež uporabnikov. Poleg omenjenega AB testiranja je priročna in pogosto uporabljana tudi namestitev v senci (angl. shadow deployment). S takšnim pristopom lahko preverimo in primerjamo uspešnost in učinkovitost novega modela z obstoječo verzijo brez, da bi posodobljen model imel vpliv na same uporabnike. Namestitev v senci namreč deluje nekoliko drugače kot AB testiranje, saj se model izvaja vzporedno s trenutnim produkcijskim, na način, da se vhodni podatki pošiljajo v oba modela pri čemer se napovedi produkcijskega posredujejo uporabniku, napovedi modela v senci pa se le shranijo za kasnejšo analizo in primerjavo. S tem lahko dobimo realen vpogled v delovanje posodobljenega napovednega modela, brez da bi vplival na delovanje produkcije [12].

2.5. Nadzorovanje

Nadzorovanje je v kontekstu napovednih modelov namenjenih sledenju delovanja modelov v produkcijskih okoljih, kjer se spremljajo vrednosti metrik napovedne uspešnost kot tudi propustnost ter latence pri pridobivanju napovedi. Slednje so ključne za zgodnjo zaznavanje ozkih grl, ki so pogost pojav predvsem pri kompleksnih modelih globokega učenja. Na ta način se nam tudi omogoči vpogled v realno časovne vrednosti metrik posameznega modela, s pomočjo česar lahko hitreje zaznamo tudi prisotnost spremembe porazdelitev vrednosti vhodnih podatkov. Dodatno nam neprekinjeno nadzorovanje omogoča tudi hitrejšo zaznavanje morebitnega poslabšanja napovedne uspešnosti modelov ali zaznavo prisotnih nepravilnosti, kar nam omogoča pravočasno posredovanje in prilagoditev napovednega modela [7].

3 Validiranje, testiranje in nadzorovanje podatkov

Validiranje, testiranje in nadzorovanje podatkov so ključni procesi v vsakem MLOps življenjskem ciklu. Uspešnost teh je namreč ključna za uspešnost učenja napovednega modela kot tudi kasnejšo napovedno uspešnost. V splošnem validacija podatkov predstavlja preverjanje same strukture podatkovne zbirke ter zalog vrednosti posameznih značilnic. Na drugi strani testiranje v splošnem služi za namen identifikacije potencialne spremembe v porazdelitvi vrednosti vhodnih značilnic. Morebitna nezaznava spremembe v porazdelitve vrednosti vhodnih značilnic lahko prav tako ključno vpliva na napovedno uspešnost modela v produkcijskem okolju.

3.1. Validiranje podatkov

Obstajajo različni pristopi validacije podatkov, ki jih v splošnem delimo na validacijo znotraj paketa podatkov (angl. Single-batch) in validacijo med paketi podatkov (angl. Inter-batch). Validacija znotraj paketa se uporablja za naslavljanje problema anomalij v sklopih podatkov. Cilj v tej fazi je zaznati anomalijo in obvestiti odgovorne osebe o napaki in začeti z preiskavo. Na drugi strani pa je validacija med paketi namenjena za preverjanje obstoja signifikantnih razlik med različnimi sklopi ali verzijami podatkov [15]. Za preverjanje anomalij v podatkih se tipično

uporablja pristop validacije na podlagi sheme. Sistem preverja strukturo novih podatkov s predpisano shemo, ki služi kot logični model za podatke. V kolikor se pojavijo kakršne koli razlike od pričakovane strukture, so podatki označeni kot anomalija in se sproži opozorilo, ki zahteva intervencijo [15]. Dodatno se validacija med paketi uporablja tudi za namen iskanja napak, ki nastanejo zaradi spremembe značilnic, ki so lahko posledica spremembe v okolju.

3.2. Testiranje podatkov

Sčasoma je pričakovano, da se napovedna uspešnost modelov poslabša (degradira), kar lahko pripišemo dejstvu, da je večina modelov strojnega učenja zgrajena nad zgodovinskimi podatki, okolje v katerem napovedni model deluje pa se konstanto spreminja. Posledično pride do pojava, ki ga imenujemo sprememba porazdelitve vrednosti značilnic. Poznamo tri glavne vrste takšnih sprememb in sicer kovariatno spremembo (angl. Covariate shift), spremembo predhodnih verjetnosti (angl. Prior probability shift) ter konceptualno spremembo (angl. Concept shift). Kovariatna sprememba označuje spremembe distribucij vhodnih podatkov oziroma značilnic. Sprememba predhodnih verjetnosti predstavlja spremembo distribucije izhodnih podatkov oziroma spremenljivk. Konceptualna sprememba pa predstavlja spremembo statistične povezave med vhodnimi in izhodnimi podatki [13].

Kovariatna sprememba se navezuje na spremembe v distribuciji vrednosti vhodnih podatkov. Definirana je na več različnih načinov, bistveno pa je, da se skozi čas populacija spreminja in z njo vrednosti oziroma porazdelitve vhodnih podatkov. Govorimo torej o pojavu, ki nastane kadar pride do večje razlike med podatki, ki so bili uporabljeni za učenje modela, in podatki, ki se uporabljajo za ustvarjanje napovedi [13].

Sprememba predhodnih verjetnosti predstavlja pojav, v katerem se porazdelitev ciljnih razredov med učenjem napovednega modela in samo uporabo modela v produkciji spremeni. To pomeni, da so značilnosti posameznih razredov v učni podatkovni zbirki in vhodnimi podatki v produkciji enake, vendar se je delež posameznih razredov med tema dvema fazama spremenil. Tak pojav lahko pomembno vpliva na delovanje modela v praksi, saj večina nadzorovanih modelov implicitno predpostavlja, da so podatki v učnem in testnem okolju iz iste porazdelitve. Če ta predpostavka ne drži, so lahko napovedi sistematično pristranske [13].

Konceptualna sprememba se zgodi, ko se razmerje med vhodnimi značnilnicami in ciljno značilnico spremeni, kar predstavlja najtežji izziv med predstavljenimi tipi podatkovnih sprememb. V takšnem primeru ostajajo porazdelitve vrednosti značilnic enake, tako pri vhodnih kot izhodnih značilnicah, spremeni se pa funkcija preslikave iz vhodnih v izhodne vrednosti. Takšna sprememba pomeni, da enaki vhodni podatki skozi čas vodijo do različnih izhodov. Na primer, v sistemu za zaznavanje goljufij se lahko vedenjski vzorci uporabnikov ne spremenijo, vendar lahko nova oblika zlorabe povzroči, da isti vzorec, ki je bil v preteklosti označen kot legitimen, v sedanjosti predstavlja goljufijo. Konceptualne spremembe so posebej problematične zato, ker jih ni mogoče preprosto zaznati zgolj z analizo porazdelitev vrednosti značilnic. Zahtevajo stalno nadzorovanje delovanja napovednega modela v realnem okolju in pogosto vključujejo uporabo detekcijskih mehanizmov, temelječih na naprednejših statističnih testih ali periodično ponovno učenje modela. Če takšnih sprememb ne zaznamo pravočasno, lahko pride do občutnega poslabšanja kakovosti napovedi in s tem do napačnih odločitev, kar je še posebej kritično v varnostno občutljivih ali reguliranih okoljih.

3.3. Nadzorovanje podatkov

Sprememba v porazdelitvi vrednosti značilnic se lahko zazna, kadar se statistične značilnosti vhodnih podatkov razlikujejo od porazdelitve vrednosti učnih podatkov. Tipično jih lahko zaznamo že na podlagi razlik v povprečnih vrednostih značilnic, variance ali oblike porazdelitve vhodnih podatkov. Zaznava takšnih pojavov zahteva neprekinjen nadzor in statistično analizo novih podatkov v primerjavi z zgodovinskimi podatki [12]. Za zaznavo

tovrstnih sprememb v realnem času poznamo različne tehnike, kot so statistične metode, tehnike procesiranja z uporabo UI mehanizmov, metrike na podlagi razdalje in vizualizacijske tehnike [14].

V ta namen pogosto uporabljamo statistične teste kot so Kolmogorov-Smirnov test, indeks populacijske stabilnosti (PSI) in Chi-kvadrat test (angl. Chi-Square Test). Kolmogorov-Smirnov test se uporablja za preverjanje enakosti dveh distribucij populacij, z namenom ugotovitve ali sta populaciji pomembno različni. Indeks populacijske stabilnosti se uporablja za ustvarjanje statistike, ki prikazuje relativno gibanje vrednosti značilnic v in med razredi. Ko je PSI vrednost višja od 0,25, potem so zaznane velike ravni spremembe v porazdelitvi podatkov. Chi-Square test se uporablja v primeru kategoričnih spremenljivk. Testira porazdelitve opazovanih vrednosti z že znanimi oziroma pričakovanimi vrednostmi [14].

Poleg omenjenih statističnih metod se pogosto uporabljajo tudi metode, ki delujejo na podlagi razdalje, kot na primer Kullback-Leiblerjeva (KL) divergenca in Jensen-Shannonova divergenca. Kullback-Leiblerjeva divergenca razloži do kakšne mere se razlikuje ena porazdelitev verjetnosti od druge. Jensen-Shannonova divergenca pa je zrcalna slike KL divergence [14].

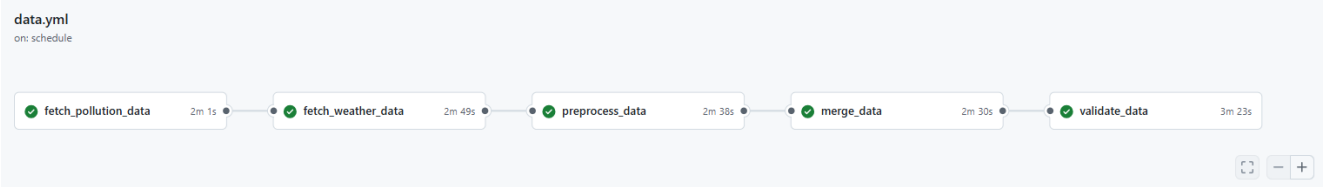
V procesu MLOps je ključno, da se tovrstna testiranja implementirajo v cevovod in izvajajo ob vsaki posodobitvi podatkov, ki jih uporabljamo za učenje napovednih modelov ter na tak način preprečimo morebitno spremembo porazdelitve vrednosti podatkov, ki bi lahko negativno vplivala na napovedno uspešnost modela v produkciji.

4 Študija primera: Airly

Airly je na MLOps pristopu temelječa rešitev, ki omogoča ogled napovedi kakovosti zraka v Sloveniji. Natančneje, omogoča pregled trenutnega stanja kot tudi napovedi za delce PM10 in PM2.5. Podatki se kontinuirano zbirajo in prikazujejo za 21 meteoroloških postaj na urnem intervalu. Poleg vpogleda v trenutno stanje in napovedi, rešitev omogoča tudi nadzorovanje delovanja napovednih modelov, kjer so prikazane metrike napovedne uspešnosti za posamezni model in primerjave med napovedanimi ter dejanskimi vrednostmi delcev v zraku za vsako meteorološko postajo. Neprekinjeno zbiranje, validiranje in testiranje podatkov nam omogoča zaznavanje sprememb v porazdelitvah vrednosti podatkov, na podlagi katerih lahko potem sprejmemo odločitev ali model zamenjamo z novim ali ne. Dodatno je neposredno podprto tudi upravljanje z vsemi modeli v sklopu rešitve.

4.1. Cevovod za podatke

Vsaka MLOps rešitev vsebuje različne cevovode za izvedbo nalog. Takšen pristop je uporabljen tudi v naši rešitvi. MLOps proces Airly-ja se začne s cevovodom za podatke (Slika 2), ki jih pridobimo iz različnih virov in jih nato predprocesiramo (shranimo podatke na urni interval, odstranimo duplikate). Sledi združevanje podatkov, kjer združimo vrednosti PM delcev z vremenskimi podatki. Zatem sledi validacija in testiranje podatkov z uporabo statističnih testov. Če so testi uspešno prestani, se cevovod nadaljuje z razdelitvijo podatkov na učno in testno množico. V našem primeru se ta cevovod izvaja periodično in sicer enkrat na dan.



Slika 2: Grafični prikaz podatkovnega cevovoda.

Podatki se shranjujejo in verzionirajo z uporabo orodja DVC, ki omogoča shranjevanje velike količine podatkov in razbremenitev orodja za verzioniranje kode, v našem primeru orodja Git. V vsakem stanju cevododa se le ti dodajo na oddaljen strežnik s pomočjo DVC ukazov, ki preverja in beleži spremembe v posameznem direktoriju.

V kolikor spremembe obstajajo se za direktorij generira nova datoteka oziroma posodobi obstoječa, kamor se zapiše meta podatke, kot so število datotek, vrednost zgoščevalne funkcije (angl. hash), velikost direktorija, s čimer je enostavno omogočeno sledenje verzijam podatkov. Nato se spremembe naložijo na oddaljen DVC podatkovni strežnik, spremenjena datoteka z metapodatki pa se objavi na Git strežnik.

4.2. Cevovod za učenje modelov

Ker se zavedamo, da lahko pride do spremembe v porazdelitvi podatkov, smo razvili tudi avtomatiziran cevovod za učenje modelov. Cevovod se trenutno izvaja periodično in sicer enkrat mesečno ali pa na zahtevo. Najprej se pridobi koda celotnega repozitorija skupaj z ».dvc« datotekami, ki vsebujejo potrebne metapodatke, zatem pa se pridobijo najnovejše verzije podatkov iz DVC strežnika, ki so uspešno prestale proces validiranja in testiranja. Šele nato se izvede dejansko proces učenja novih modelov, pri čemer so vsi koraki zabeleženi na način, da lahko zagotovimo reprodukcijo eksperimentov. Ko so modeli naučeni, se nova verzija, skupaj z metapodatki in trenutno verzijo podatkov, shrani na MLFlow strežnik in se ji dodeli nova verzija.

4.3. Nadzorovanje v produkciji

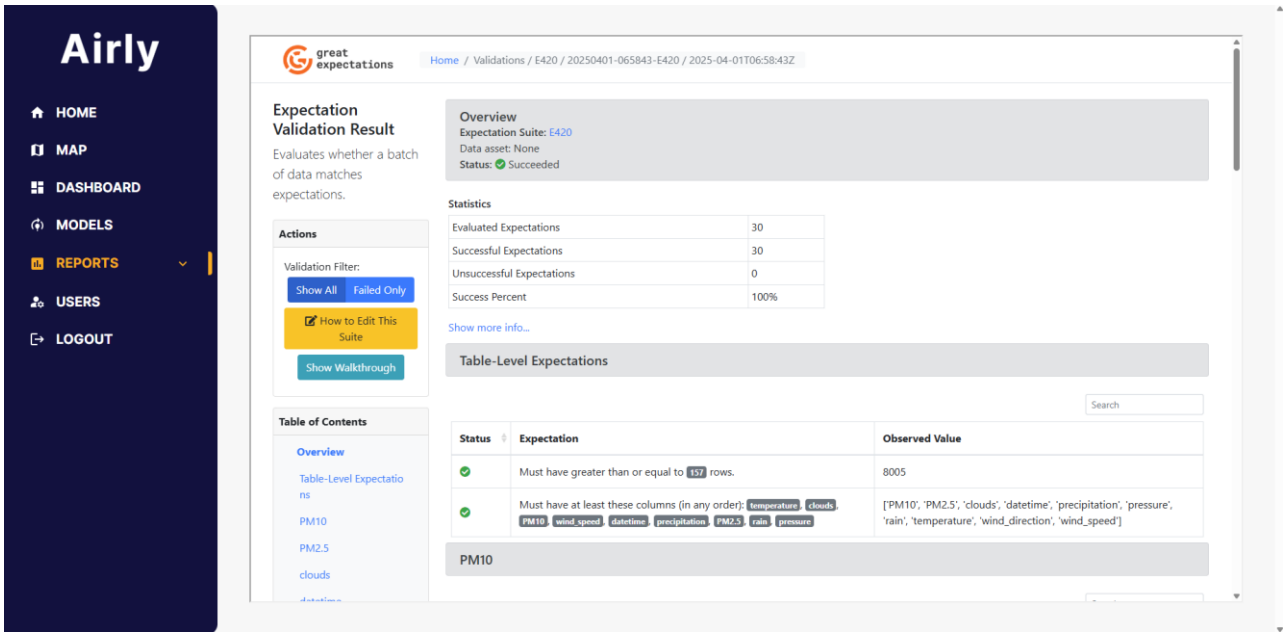
Za vsako postajo si lahko ogledamo napovedane vrednosti s strani različnih modelov ter dejanske vrednosti delcev PM v zraku. Slika 3 prikazuje nadzorno ploščo, ki omogoča vpogled v uspešnost delovanja napovednega modela. Siva krivulja predstavlja realne vrednosti delcev PM10 in PM2.5 za izbrano meteorološko postajo, medtem ko črna krivulja predstavlja napovedane vrednosti. Dodatno ima platforma implementirano še zmožnost testiranja napovednega modela v produkciji z uporabo pristopa testiranja v senci. V našem primeru rumena krivulja predstavlja model v senci, katerega napovedi uporabniki ne vidijo, nam pa služi za namen primerjave napovedne uspešnosti s trenutnim produkcijskim napovednim modelom.

Slika 3: Prikaz pogleda za nadzorovanje napovedi.

Poleg vpogleda v primerjave napovedanih vrednosti onesnaženja zraka, rešitev omogoča tudi vpogled v vrednosti izbranih regresijskih metrik s katerimi lahko kvantitativno ocenimo napovedno uspešnost modela. Slika 4 prikazuje vpogled v vrednosti regresijskih metrik povprečna absolutna napaka (angl. Mean absolute error, MAE) ter povprečna kvadratna napaka (angl. Mean squared error, MSE) za izbrano meteorološko postajo.

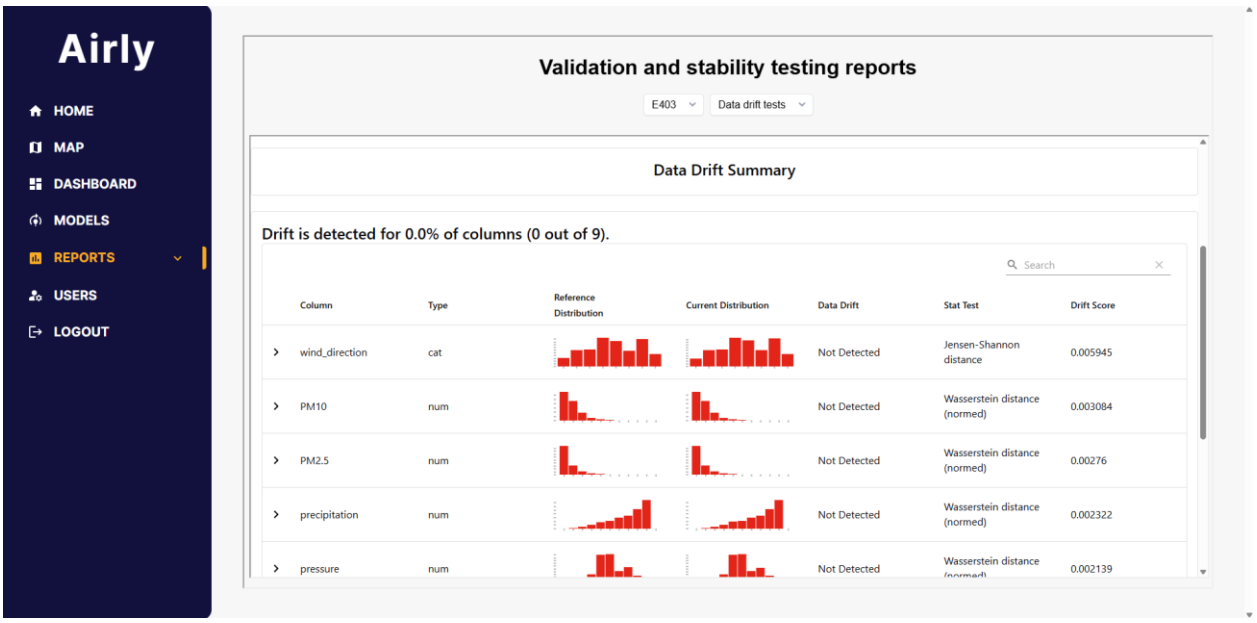
Slika 4: Prikaz pogleda vrednosti regresijskih metrik.

V sklopu nadzorovanja si lahko tudi ogledamo poročila o izvedenih procesih validacije in testiranja podatkov. V poročilu o validaciji podatkov (Slika 5) lahko vidimo število vseh vrstic, stolpcev, zalogo vrednosti posameznega stolpca ter koliko vrednosti je bilo nepričakovanih oziroma odstopalo od zastavljene sheme. Prav tako pa imamo omogočen vpogled v vsa zgodovinska validacijska poročila.



Slika 5: Primer vpogleda v validacijsko poročilo.

V poročilu o testiranju podatkov (Slika 6) lahko preverimo, ali je prišlo do spremembe porazdelitve vrednosti podatkov za katero izmed značilnic, vrsto podatkov, referenčno distribucijo podatkov, trenutno distribucijo podatkov, kateri statistični test je bil uporabljen za testiranje podatkov ter dejanski rezultat spremembe porazdelitve podatkov. Poleg pregleda zadnje aktualnega poročila o testiranju imamo tudi vpogled v vsa predhodna poročila o testiranju.



Slika 6: Primer vpogleda v poročilo testiranja podatkov.

Vpogled v metrike napovedne uspešnosti modelov, kot tudi v poročila o validaciji in testiranju podatkov, nam omogočajo celovito razumevanje delovanja napovednega modela. S tem lahko ugotovimo, kako dobro se model generalizira na še nevidene podatke, kako občutljiv je na odstopanja v vhodnih podatkih ter ali obstajajo znaki degradacije napovedne uspešnosti skozi čas. Spremljanje teh metrik omogoča pravočasno odkrivanje težav, kot so premiki v podatkovni porazdelitvi, zmanjšana točnost napovedi ali porast neustreznih vhodov, kar je ključno za zagotavljanje stabilnosti in zanesljivosti produkcijskega modela. Takšen vpogled je temelj za informirano odločanje o ponovnem učenju modela, prilagoditvah cevovoda za obdelavo podatkov ali izboljšavah validacijskih pravil.

5 Zaključek

Vzpostavitev kakovostne MLOps rešitve zahteva premišljeno uporabo širokega nabora orodij in tehnologij, ki morajo med seboj učinkovito sodelovati. Najpogosteje uporabljena orodja so odprtokodne rešitve, kot so MLFlow za sledenje poskusom in verzioniranje modelov, DVC za upravljanje z različicami podatkov in cevovodov ter TensorFlow za gradnjo in učenje modelov. Vendar pa tehnična kompleksnost integracije teh orodij pogosto zahteva poglobljeno razumevanje posameznih komponent ter njihovo natančno uskladitev znotraj celotne arhitekture. Vzpostavitev takšne infrastrukture ni trivialna naloga. Vključuje tako načrtovanje ustreznega podatkovnega cevovoda kot tudi zagotavljanje zadostnih računalniških virov za izvajanje učenja napovednih modelov. MLOps ni enkraten projekt, temveč živ proces, ki se mora prilagajati novim podatkom in spreminjajočim se okoliščinam. Zato je ključno, da se podatki neprekinjeno nadzorujejo, saj lahko že manjše spremembe v vhodnih podatkih ali strukturi povzročijo odstopanja, ki lahko ključno vplivajo na delovanje modela.

Kljub številnim prednostim pa MLOps prinaša tudi nekatere izzive. Integracija odprtokodnih orodij lahko naleti na težave zaradi nepopolne dokumentacije, pomanjkljive podpore ali nedoslednosti med knjižnicami. Poleg tega so potrebe po računski moči in podatkovni shrambi lahko visoke, še posebej v okoljih z obsežnimi podatki in kompleksnimi modeli. Vse to povečuje časovno kompleksnost za vzpostavitev delujoče in stabilne infrastrukture.

Kljub relativno zahtevni vzpostavitvi pa dobro implementirana MLOps rešitev prinaša pomembne dolgoročne koristi saj omogoča zanesljivo in ponovljivo razvijanje, nameščanje ter vzdrževanje napovednih modelov v produkcijskih okoljih. S tem ne le dviguje kakovost napovednih modelov, temveč tudi omogoča njihovo proaktivno spremljanje, lažje odpravljanje napak in hitrejšo prilagoditev na spremembe. MLOps tako postaja ključen gradnik sodobnih inteligentnih sistemov, kjer je zaupanje v delovanje modela enako pomembno kot njegova napovedna uspešnost.

Literatura

[1] Jameel Francis, Forbes, „Forbes,“ 15 11 2024. [Elektronski]. Dostopno na: https://www.forbes.com/councils/forbestechcouncil/2024/11/15/why-85-of-your-ai-models-may-fail/.

[2] E. Breck, S. Cai, E. Nielsen, M. Salib in D. Sculley, „The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction,“ v Proceedings of IEEE Big Data, 2017.

[3] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy in A. Bouchachia, „A Survey on Concept Drift Adaptation,“ ACM Computing Surveys, Izv. 1, št. 1, 2013.

[4] J. Kazmierczak, K. Salama in V. Huerta, „MLOps: Continuous delivery and automation pipelines in machine learning,“ 28 08 2024. [Elektronski]. Dostopno na: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

[5] S. Wazir, G. S. Kashyap, in P. Saxena, „MLOps: A Review“, 2023. [Elektronski]. Dostopno na: https://arxiv.org/abs/2308.10908

[6] M. Treveil idr., Introducing MLOps. O'Reilly Media, 2020.

[7] S. K. Thota, V. R. R. Alluri, V. Kumar, V. K. R. Vangoor, in S. Chitta, „MLOps: Streamlining Machine Learning Model Deployment in Production“, Journal of Artificial Intelligence, let. 2, str. 186–205, apr. 2022.

[8] E. Orr, „Data Version Control: What It Is and How It Works“. Pridobljeno: 23. marec 2025. [Na spletu]. Dostopno na: https://lakefs.io/data-version-control/

[9] Y. Wu, E. Dobriban, in S. Davidson, „DeltaGrad: Rapid retraining of machine learning models“, v Proceedings of the 37th International Conference on Machine Learning, H. D. III in A. Singh, Ur., v Proceedings of Machine Learning Research, vol. 119. PMLR, apr. 2020, str. 10355–10366. [Na spletu]. Dostopno na: https://proceedings.mlr.press/v119/wu20b.html

[10] A. Mahadevan in M. Mathioudakis, „Cost-Effective Retraining of Machine Learning Models“, 2023. [Na spletu]. Dostopno na: https://arxiv.org/abs/2310.04216

[11] K. Rahmani idr., „Assessing the effects of data drift on the performance of machine learning models used in clinical sepsis prediction“, Int J Med Inform, let. 173, str. 104930, 2023, doi: https://doi.org/10.1016/j.ijmedinf.2022.104930.

[12] A. Balasubramanian, „End-to-end model lifecycle management: An MLOPS framework for drift detection, root cause analysis, and continuous retraining“.

[13] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V Chawla, in F. Herrera, „A unifying view on dataset shift in classification“, Pattern Recognit, let. 45, št. 1, str. 521–530, 2012, doi: https://doi.org/10.1016/j.patcog.2011.06.019.

[14] N. Kodakandla, „Data drift detection and mitigation: A comprehensive MLOps approach for real-time systems“, International Journal of Science and Research Archive, let. 12, str. 3127–3139, apr. 2024, doi: 10.30574/ijsra.2024.12.1.0724.

[15] N. Polyzotis, M. Zinkevich, S. Roy, E. Breck, in S. Whang, „Data validation for machine learning“, Proceedings of machine learning and systems, let. 1, str. 334–347, 2019.