Tema: Uvod u strojno učenje

Kolegij: Strojno učenje za primjene u okolišu

Opis lekcije:

Ova uvodna lekcija daje osnovni pregled strojnog učenja (ML) i njegove uloge u rješavanju okolišnih i ekoloških problema. U lekciji se uvode glavni ML zadaci (klasifikacija, regresija, klasteriranje) i strategije učenja (nadzirano i nenadzirano učenje). Naglasak je na razumijevanju ključnih koncepata, uključujući pripremu podataka, inženjering značajki, treniranje modela i evaluaciju, uz jasne vizualne primjere. Problem podnaučenosti i prenaučenosti te njihove posljedice objašnjavaju se detaljno. Okolišne primjene uključuju klasifikaciju razina kvalitete zraka i detekciju staništa.

Ishodi učenja:

Nakon završetka ove nastavne jedinice studenti će moći:

1. Objasniti što je ML i opisati njegove glavne primjene u znanosti o okolišu.
2. Razlikovati nadzirano i nenadzirano učenje.
3. Prepoznati osnovne ML zadatke kao što su regresija, klasifikacija i klasteriranje.
4. Razumjeti osnovni tijek rada ML projekta (priprema podataka, treniranje, evaluacija).
5. Prepoznati i objasniti česte probleme u učenju ML modela, kao što su prenaučenost, podnaučenost i problemi kvalitete podataka.
6. Trenirati i evaluirati jednostavan ML model pomoću okolišnih podataka.

Uvod

Strojno učenje (ML) omogućuje računalima učenje iz podataka bez potrebe za eksplicitnim programiranjem. Posljednjih je godina ML našao široku primjenu u rješavanju različitih okolišnih problema, nadopunjujući ili, u nekim slučajevima, zamjenjujući tradicionalne pristupe modeliranju okoliša koji se ponajprije temelje na formalnim fizikalnim i kemijskim jednadžbama, kao i na njihovoj kombinaciji s drugim formalizmima u biologiji, geologiji i ekologiji. S povećanjem količine i raznolikosti okolišnih podataka (senzori, IoT uređaji, sustavi monitoringa velikog opsega), fokus se pomaknuo prema metodama vođenima podacima koje uče izravno iz prikupljenih podataka.

Strojno učenje disciplina je vođena podacima i jedno od najvažnijih područja umjetne inteligencije. Umjesto propisivanja fiksne strukture modela, ML algoritmi izvode zaključke o temeljnim odnosima iz primjera, čime znanstvenicima i stručnjacima omogućuju modeliranje sustava u kojima su eksplicitni fizikalni zakoni nepoznati, nepotpuni ili previše složeni.

Okolišni fenomeni (npr. klimatski procesi, degradacija ekosustava i dinamika onečišćenja) prirodno su složeni, nelinearni i višedimenzionalni problemi. Područje strojnog učenja nudi različite alate i pristupe koji mogu odgovoriti na takve složene izazove. Štoviše, budući da su ti problemi popraćeni bogatim podacima, često ih je moguće učinkovitije obraditi ML-om nego tradicionalnim regresijskim ili simulacijskim modelima.

Neki primjeri primjene ML-a u znanostima o okolišu:

- Predviđanje klime i vremena: ML modeli prepoznaju obrasce u dugoročnim meteorološkim zapisima i satelitskim opažanjima kako bi predvidjeli temperaturne anomalije ili ekstremne oborine .
- Modeliranje kvalitete zraka i vode: prediktivni modeli uče korelacije između emisija, meteoroloških varijabli i koncentracija onečišćivala .
- Daljinska istraživanja i klasifikacija korištenja zemljišta: ML modeli temeljeni na slikama automatski klasificiraju tipove zemljišnog pokrova iz multispektralnih satelitskih podataka .
- Praćenje bioraznolikosti i ekosustava: ML pomaže u modeliranju rasprostranjenosti vrsta, akustičkoj klasifikaciji i detekciji ekoloških anomalija .

Već smo spomenuli da je strojno učenje sustav koji uči iz podataka bez potrebe za eksplicitnim programiranjem za zadatak koji treba riješiti.

> *„Za računalni se program kaže da uči iz iskustva E s obzirom na neku klasu zadataka T i mjeru uspješnosti P ako se njegova uspješnost u zadacima iz T, mjerena s P, poboljšava s iskustvom E.”*

Ova formulacija naglašava tri ključne komponente:

- Podaci (iskustvo E): temelj iz kojega model uči obrasce.
- Zadatak (T): ono što se od modela očekuje da izvrši — primjerice klasificiranje, predviđanje ili klasteriranje.
- Uspješnost (P): kvantitativna mjera koliko dobro model obavlja zadatak (točnost, RMSE, F1-mjera itd.).

Primjer: želimo razviti aplikaciju koja može detektirati automobile na slikama. Kada bismo željeli tvrdo kodirati skup pravila (IF-THEN) koji opisuje automobil, primjerice:

AKO slika prikazuje 4 kotača, pravokutnog je oblika i ima vjetrobransko staklo, tada je to AUTOMOBIL; u suprotnom, NIJE AUTOMOBIL.

Takav pristup brzo ne uspijeva jer automobili izgledaju različito iz različitih kutova, osvjetljenje utječe na prepoznavanje, a neki objekti nalikuju automobilima. Stoga za ovaj pristup kažemo da je previše rigidan i da ne daje dobre rezultate.

Iz više smo izvora čuli da pristup strojnog učenja može riješiti takav tip problema, uz relativno veliku količinu podataka.
Stoga smo odlučili primijeniti tu strategiju: prikupili smo pristojan broj slika automobila u različitim situacijama i jednak broj slika koje ne sadrže automobile. Za taj smo zadatak pročitali da se preporučuje određeni ML model za prepoznavanje slika. Sve su slike definirane odgovarajućim i točnim oznakama (CAR/NO CAR). Ovdje je zadatak ML modela predvidjeti ispravnu oznaku iz slike. Taj se zadatak uči iz slika (podataka), odnosno možemo reći da model uči iz iskustva (prikazivanjem slika). Na početku će model u mnogim slučajevima griješiti, ali s vremenom (pokazivanjem više podataka – iskustva) naučit će ispravno predviđati oznake u većini slučajeva. Postotak ispravno predviđenih oznaka slika mjera je uspješnosti koja pokazuje koliko je dobro model naučio zadatak.

Kaže se da model uči zato što se njegova uspješnost u detekciji automobila na slikama poboljšava kako stječe više iskustva s podacima!

Tipičan ML pipeline uključuje:

1. Prikupljanje i predobradu podataka
2. Inženjering značajki i reprezentaciju
3. Odabir i treniranje modela

4. Evaluaciju i validaciju
5. Interpretaciju i primjenu

Osim pouzdane prediktivne uspješnosti u znanosti o okolišu, ML pruža osnovu za visokokvalitetne sustave odlučivanja u uvjetima nesigurnosti , nudeći interpretabilne uvide u važnost značajki , prostorne ovisnosti i osjetljivost sustava . Kako okolišni podaci nastavljaju eksponencijalno rasti, ML pismenost postaje nužna sastavnica suvremenog upravljanja okolišnim podacima .

Ispod površine temeljnih pojmova ML-a

Znanost o okolišu u svojoj je biti bogata velikom količinom podataka iz heterogenih izvora, pa je prirodno iskoristiti korisnost ML modela u tom području. ML nadopunjuje konvencionalne znanstvene pristupe identificiranjem skrivenih obrazaca i izvođenjem korelacija. U ovom uvodnom predavanju fokus nije na detaljnim objašnjenjima modela za različite ML zadatke, što će biti obrađeno u sljedećim predavanjima, nego na temeljnim načelima koja vrijede za sve zadatke. Glavni je cilj konceptualno razumijevanje, a ne matematički formalizam koji stoji iza rada ML modela; taj će formalizam biti obrađen u predavanjima posvećenima specifičnim ML zadacima.

Nadzirano i nenadzirano učenje

Iz definicije strojnog učenja vidimo da ML modeli mogu naučiti obavljati zadatak učenjem iz podataka. Postoje dvije osnovne vrste učenja: nadzirano i nenadzirano. Nadzirano učenje strategija je učenja u kojoj se model trenira na podacima s poznatim ulaznim vrijednostima i pripadajućim izlazima. Primjer je klasifikacija satelitskih slika u osam kategorija zemljišnog pokrova: izgrađena područja, ceste, vodene površine, poljoprivredno zemljište, grmlje, šuma, travnjaci i ostalo . Oznake kategorija dobivaju se od domenskih stručnjaka, pa unaprijed znamo gdje se nalazi anotacija za svaku sliku u skupu podataka; model to, međutim, ne zna. Primjenom odgovarajuće mjere uspješnosti (npr. apsolutne točnosti), model može naučiti predviđati ta područja na označenim slikama i, što je još važnije, na novim, prethodno neviđenim slikama. Naziv nadzirano ovdje se odnosi na anotacije (oznake) dobivene od nadzornika, domenskih stručnjaka koji služe kao temeljna istina za učenje zadatka. Osim spomenutog zadatka klasifikacije, drugi tipičan zadatak nadziranog učenja jest regresija, u kojoj ML model uči iz podataka pomoću ulaznih vrijednosti s poznatim realno vrijednosnim izlazima. Primjer je predviđanje koncentracije nitratnog dušika i klorida iz skupa neovisnih varijabli (prosječni dnevni površinski protok, 24-satna oborina prije trenutka uzorkovanja, lokacija mjesta uzorkovanja, pokrov usjeva, temperatura, pH, otopljeni kisik, oksidacijsko-redukcijski potencijal i električna vodljivost). Ovisne varijable pokazatelji su kvalitete površinskih voda u poljoprivrednim slivovima . U regresijskim zadacima nadzor je neizravan, izražen kroz prikupljene neovisne i ovisne podatke.

S druge strane, nenadzirano učenje nema unaprijed dobro definiran, poznat izlaz za promatrane ulaze. Zapravo su poznati samo ulazi. ML model mora pronaći pravilnosti u ulaznim podacima i istaknuti smislene obrasce s konkretnim značenjem u području interesa. Ovdje ne vidimo nikakav utjecaj nadzornika koji bi definirao temeljnu istinu; zbog toga je ova vrsta učenja u području ML-a poznata kao nenadzirano učenje.

Tipičan zadatak u paradigmi nenadziranog učenja jest klasteriranje, pri čemu je cilj grupirati slične podatkovne instance u istu skupinu i osigurati da svaka različita instanca pripada drugoj skupini (tzv. crisp clustering pristup), dok u slučajevima u kojima ista instanca može pripadati više od jedne skupine (klastera) s pridruženom vrijednošću pripadnosti govorimo o soft clustering pristupima. U znanosti o okolišu klasteriranje ima važnu ulogu i može se primijeniti za procjenu kvalitete podzemnih voda .

Postoji mnogo ML modela koji se mogu primijeniti za navedene zadatke:

- Klasifikacija (SVM – Support Vector Machines, stabla odluke, RIPPER, K-NN, umjetne neuronske mreže, logistička regresija, Random Forest, Naive Bayes, AdaBoost, Bagging, Stacking itd.)
- Regresija (linearna regresija, SVM za regresiju, stabla odluke za regresiju, višeslojni perceptron itd.)
- Klasteriranje (K-means clustering, Expectation-maximization, DBSCAN, HDBSCAN, Cobweb, hijerarhijsko klasteriranje, Fuzzy C-means itd.)

Većinu ovih ML modela detaljnije ćete učiti u sljedećim predavanjima, pa ćemo ovdje dati samo osnovni uvid u model potreban za daljnje izlaganje osnovnih koncepata u ML-u, kao što su priprema podataka, inženjering značajki, treniranje, testiranje, evaluacija uspješnosti te fenomeni prenaučenosti i podnaučenosti. Dodatno će biti predstavljen kratak uvod u Google Colab.

Logistička regresija – temeljni uvid

Iako u nazivu sadrži riječ regresija, logistička regresija klasifikacijski je model koji se najčešće koristi za probleme binarne klasifikacije. Takva klasifikacija, kao što naziv sugerira, modelira problem s dvije klase, tipično označene s 0 i 1.

Cilj joj nije predvidjeti kontinuiranu vrijednost, nego procijeniti vjerojatnost da primjer pripada određenoj klasi (najčešće klasi 1).

Formalno, modelira:

$P(y=1;x,\theta)$ Logično, kada znamo ovu vjerojatnost, poznata je i druga:

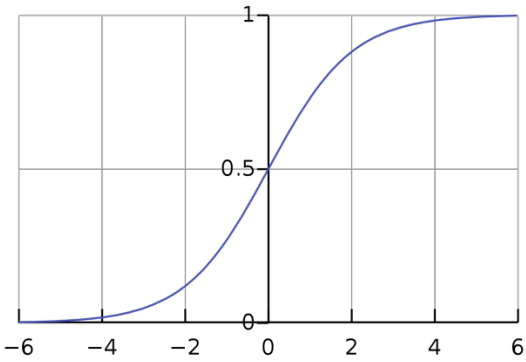
$$P(y=0;x,\theta)=1-P(y=1;x,\theta)$$

Gdje je:

- **x vektor značajki**
- y (0 ili 1) oznaka klase
- θ συ παραμετρι μοδελα (τεžινε)

U osnovi logističke regresije nalazi se sigmoidna (logistička) funkcija, koja preslikava svaki realno vrijednosni ulaz u interval (0, 1):

$$h(x) = \frac{1}{1 + e^{-\theta^T \cdot x}}$$



Slika 1. Sigmoidna (logistička) funkcija (Izvor: https://commons.wikimedia.org/wiki/File:Logistic-curve.svg)1

To je ključno jer:

- Vjerojatnosti moraju biti između 0 i 1,
- sigmoidna funkcija glatko pretvara linearnu kombinaciju značajki () u vjerojatnost. $\theta^T \cdot x$

Interpretacija je sada snažno povezana s vjerojatnostima, a ne samo s oznakama. Logistička regresija proizvodi numerički izlaz:

- $P(y=1 \mid x)$— vjerojatnost pozitivne klase
- $P(y=0 \mid x) = 1 - P(y=1 \mid x)$ — vjerojatnost negativne klase

Zbog toga je logistička regresija osobito korisna kada:

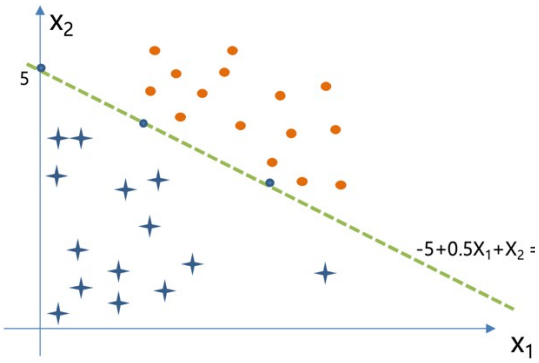
- su klase neuravnotežene,
- treba prilagoditi pragove odlučivanja,
- važni su rangiranje ili procjena pouzdanosti

Pojavljuju se dva glavna pitanja:

- Koje je načelo određivanja težina (i)?θ
- Kako izgleda granica odlučivanja ovog modela?

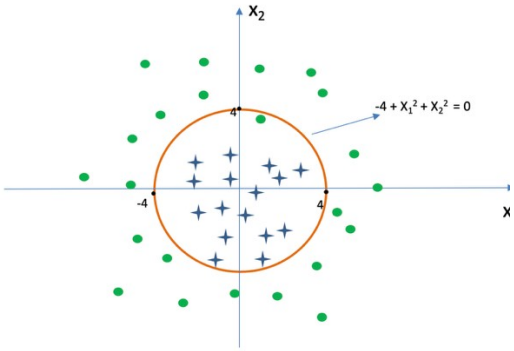
$$h(x) = ¿\ \frac{1}{1 + e^{-(\theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2)}}$$

$$h(x) = ¿\ \frac{1}{1 + e^{-(\theta_0 + \theta_1 \cdot x_1^2 \cdot + \theta_2 \cdot x_2^2)}}$$



Slika 2. Linearna granica odlučivanja

2



Slika 3. Nelinearna granica odlučivanja

3

Odluka: sve iznad linije dodjeljuje se klasi 1, inače klasi 0→

Odluka: sve izvan kruga dodjeljuje se klasi 1, inače klasi 0 →

Prvi slučaj prikazan na Slici 2 prikazuje logističku regresiju koja daje linearnu granicu odlučivanja u izvornom prostoru značajki. U drugom slučaju dobiva se nelinearna granica odlučivanja (Slika 3), ali tek nakon primjene inženjeringa značajki koji transformira izvorne ulazne značajke (npr. dodavanjem kvadratnih članova), dok model ostaje linearan u parametrima. Težine se uče iz ulaznih podataka optimizacijom odabrane mjere uspješnosti. 23θ

U ovom predavanju nije naglasak na metrikama uspješnosti niti na njihovom detaljnom objašnjenju, jer će biti obrađene u sljedećim predavanjima.

Točnost

Točnost mjeri udio ispravnih predviđanja među svim predviđanjima.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Točnost odgovara na pitanje: „Koliko je često model ukupno ispravan?”

Preciznost

Preciznost mjeri koliko je instanci predviđenih kao pozitivne doista pozitivno.

$$P = \frac{TP}{TP + FP}$$

Preciznost odgovara na pitanje: „Kada model predvidi pozitivan slučaj, koliko je često u pravu?”

Odziv (osjetljivost)

Odziv mjeri koliko je stvarnih pozitivnih instanci model ispravno detektirao.

$$R = \frac{TP}{TP + FN}$$

Visok odziv ključan je kada propuštanje okolišnog događaja (kao što je prekoračenje onečišćenja ili prisutnost staništa) nosi znatan rizik.

F1-mjera

F1-mjera kombinira preciznost i odziv u jednu mjeru pomoću njihove harmonijske sredine.

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

F1-mjera korisna je kada su važni i lažno pozitivni i lažno negativni ishodi te kada je skup podataka neuravnotežen, što je uobičajeno u okolišnim primjenama.

ROC krivulja

ROC krivulja (Receiver Operating Characteristic) prikazuje stopu istinito pozitivnih ishoda (odziv) u odnosu na stopu lažno pozitivnih ishoda za sve moguće klasifikacijske pragove.

$$FPR = \frac{FP}{FP + TN}$$

ROC krivulja ilustrira kompromis između detekcije stvarnih okolišnih događaja i pojave lažnih alarma.

ROC-AUC

ROC-AUC (Area Under the ROC Curve) sažima ROC krivulju u jednu vrijednost između 0 i 1.

Interpretacija:

- 0,5 označava nasumično pogađanje

- 1,0 označava savršeno razlikovanje

ROC-AUC odražava koliko dobro model može rangirati pozitivne slučajeve više od negativnih, neovisno o konkretnom pragu odlučivanja.

Krivulja preciznost–odziv (PR)

Krivulja preciznost–odziv (PR) prikazuje preciznost u odnosu na odziv za različite klasifikacijske pragove.

PR krivulja izravno pokazuje kako se preciznost mijenja s porastom odziva, ističući kompromis između lažnih alarma i propuštenih detekcija.

PR-AUC (prosječna preciznost)

PR-AUC sažima krivulju preciznost–odziv u jednu vrijednost. PR-AUC je osobito informativan za okolišne skupove podataka u kojima su pozitivni slučajevi rijetki, kao što su prekoračenja onečišćenja ili specifični tipovi staništa. Osjetljiviji je na lažno pozitivne ishode nego ROC-AUC.

Ključna poruka

Nijedna metrika nije univerzalno najbolja. U okolišnom strojnom učenju izbor metrike treba odražavati prioritete primjene:

• Koristite odziv kada je propuštanje stvarnih događaja rizično

• Koristite preciznost kada su lažni alarmi skupi

• Koristite F1-mjeru za uravnoteženje oboje

• Preferirajte PR-AUC i PR krivulje za detekciju rijetkih događaja

• Koristite ROC-AUC za opću usporedbu modela

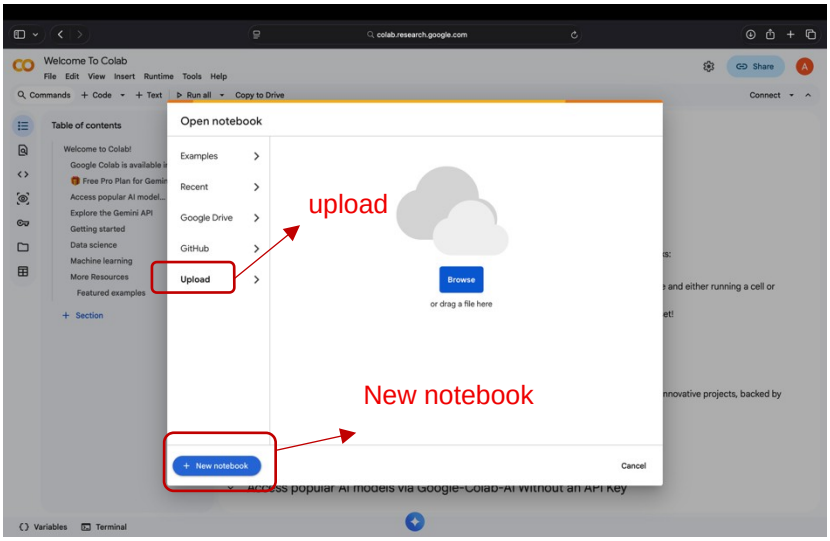
Google Colab

Google Colab platforma je u oblaku koja omogućuje besplatan pristup Googleovim računalnim resursima putem preglednika, uključujući CPU-ove, GPU-ove i TPU-ove. Temelji se na okviru Jupyter Notebook i korisnicima omogućuje pisanje i izvršavanje Python koda bez lokalnog postavljanja okruženja. Iako dolazi s unaprijed učitanim osnovnim bibliotekama za strojno učenje i besprijekorno se integrira s Google Driveom, zahtijeva Google račun i djeluje u potpunosti unutar Googleova ekosustava.

Usluga Google Colab jedan je od najjednostavnijih načina da studenti pokreću Python skripte bez stvaranja radnog okruženja sa svim potrebnim bibliotekama. Jednostavnije rečeno, omogućuje vam da na korisnički prihvatljiv način koristite okruženje za izvođenje čak i složenog Python koda koje zadovoljava sve zahtjeve i ovisnosti. Okruženje uključuje osnovni Python interpreter te dodatne biblioteke i module (unaprijed izgrađen kod koji su

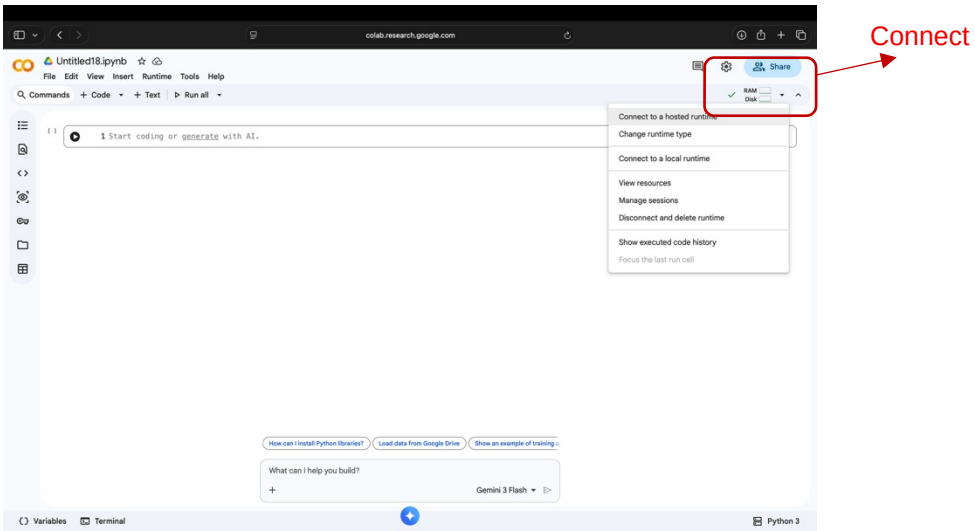
razvili i testirali pouzdani izvori radi pojednostavnjenja i ubrzanja razvoja te povećanja pouzdanosti). Te biblioteke podupiru zadatke kao što su učinkovito numeričko računanje (NumPy), robusne podatkovne strukture i analiza (Pandas), vizualizacija (Matplotlib) te suvremene implementacije strojnog učenja (Scikit-learn, TensorFlow, PyTorch).

Za korištenje Colaba potreban vam je Google račun. Ako ga nemate, možete ga izraditi ovdje. Nakon što imate račun, Colabu možete pristupiti ovdje. Trebali biste vidjeti zaslon poput ovoga:



Slika 4. Početni prozor Google Colaba4

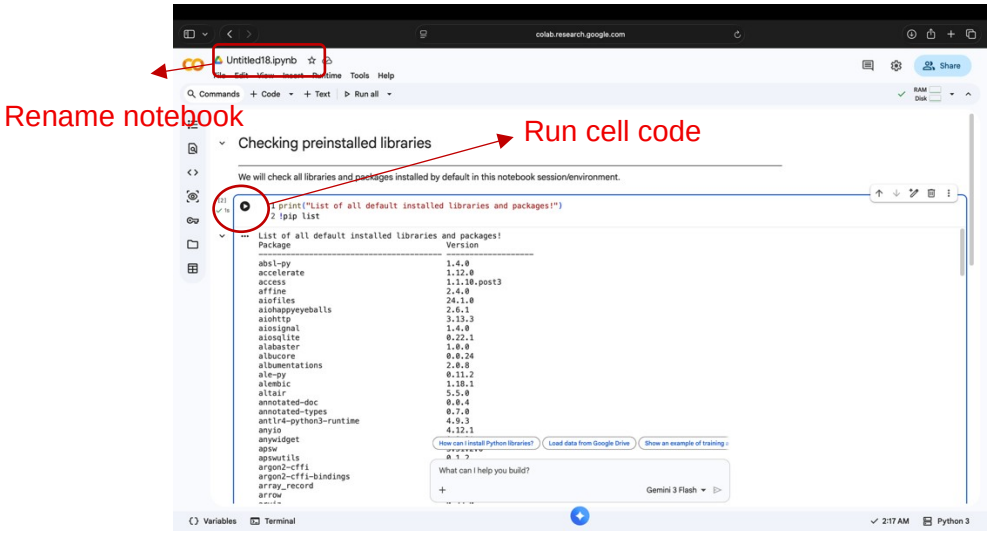
Iz ovog prozora možete otvoriti novu bilježnicu, učitati pripremljenu bilježnicu ili otvoriti postojeću. Isto se može učiniti iz izbornika File. Za sada otvorite novu bilježnicu.



Slika 5. Pokretanje nove Google Colab bilježnice5

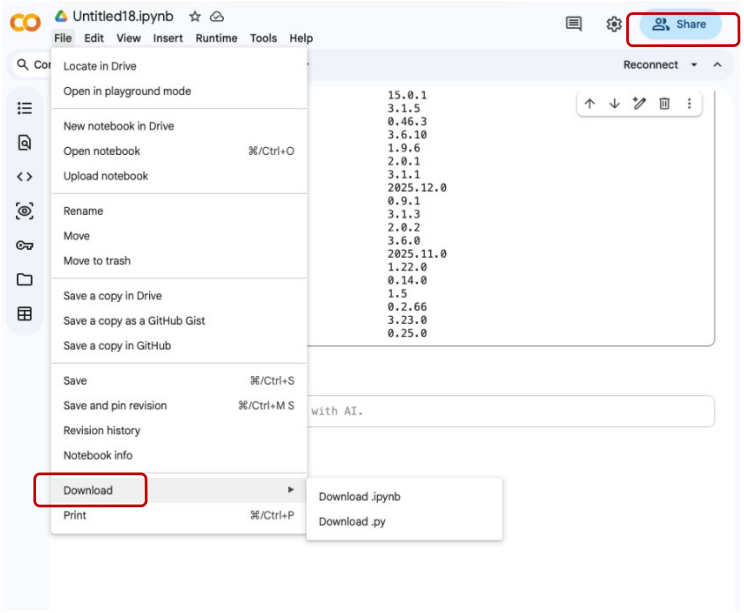
Na Slici 5, u gornjem desnom kutu kliknite na strelicu i odaberite Connect to a hosted runtime – nakon uspostave veze indikator će prikazati RAM, CPU i zelenu kvačicu.

Dostupne su dvije vrste ćelija – ćelije koda i tekstualne ćelije (markdown ćelije). Za pokretanje Python koda treba koristiti ćelije koda. Ćeliju možete izvršiti klikom na gumb „play" lijevo od ćelije ili pritiskom na „Shift+Enter" na tipkovnici, čime se odabrana ćelija automatski izvršava i umeće nova. Rad s tekstualnim ćelijama jednostavan je – umetnite tekstualnu ćeliju i dvostrukim klikom uđite u način uređivanja te počnite pisati. S desne strane bit će vidljiv pregled; nakon klika izvan ćelije prikazat će se rezultat. Pogledajte markdown vodič kako biste naučili osnove oblikovanja teksta u tim ćelijama. Preimenovanje bilježnice jednostavno je (Slika 6) promjenom naziva u gornjem lijevom kutu (gdje je ekstenzija ipynb). Slika 5. Pokretanje nove Google Colab bilježnice5Slika 6. Pokretanje Python koda u Colabu / preimenovanje bilježnice6



Slika 6. Pokretanje Python koda u Colabu / preimenovanje bilježnice6

Osim spremanja Colab bilježnice na Google Drive ili u GitHub repozitorij, bilježnicu je moguće podijeliti (u gornjem desnom kutu odaberite Share) ili je preuzeti kao .py ili .ipynb datoteku (iz izbornika File odaberite opciju Download – Slika 7.*Slika 7. Preuzimanje bilježnice / dijeljenje bilježnice7*



Slika 7. Preuzimanje bilježnice / dijeljenje bilježnice7

Predviđanje prekoračenja NO2 pomoću podataka senzora kvalitete zraka

Praćenje kvalitete zraka ima ključnu ulogu u zaštiti javnog zdravlja i podupiranju okolišne politike. Suvremene mjerne postaje kontinuirano prikupljaju velike količine podataka iz kemijskih senzora i meteoroloških instrumenata. Zbog toga je kvaliteta zraka izvrsno područje za demonstraciju kako strojno učenje može poduprijeti stvarno odlučivanje u okolišu.

U praksi se okolišne odluke rijetko temelje na predviđanju točne vrijednosti onečišćenja. Umjesto toga, ključna su pitanja obično vrlo jednostavna:

- Je li prekoračena zakonska granica kvalitete zraka?
- Treba li izdati upozorenje?
- Treba li aktivirati mjere ublažavanja?

Ovaj slučaj uporabe usmjeren je na odgovaranje na takva pitanja predviđanjem događaja prekoračenja NO_2 pomoću strojnog učenja.

SKUP PODATAKA za studiju slučaja

U ovom predavanju koristimo skup podataka Air Quality iz repozitorija UCI Machine Learning Repository, dobro poznatog javnog repozitorija skupova podataka za strojno učenje.

- **Vremensko razdoblje: ožujak 2004. do veljače 2005.**

- **Učestalost uzorkovanja: jedno mjerenje na sat**
- **Lokacija: urbana mjerna postaja u talijanskom gradu**
- **Broj opažanja: približno 9.300 satnih zapisa**

Svaki zapis predstavlja okolišne uvjete izmjerene tijekom jednog sata.

Skup podataka kombinira tri vrste mjerenja:

Koncentracije onečišćivala kao temeljna istina

Te vrijednosti dolaze iz referentnih analizatora i predstavljaju pouzdana kemijska mjerenja:

- Dušikov dioksid – NO_2(GT) (µg/m³)
- Ugljikov monoksid – CO(GT) (mg/m³)
- Dušikovi oksidi – NOx(GT) (ppb)
- Benzen – C_6H_6(GT) (µg/m³)

U opisu izvornog skupa podataka te su varijable namijenjene kao ciljne varijable za regresijske modele.

Signali senzora

Skup podataka također sadrži očitanja senzora metalnih oksida. Ti su senzori jeftiniji i šumniji od referentnih analizatora, a njihovi su signali često nelinearni i križno osjetljivi. U ovom slučaju uporabe ta očitanja senzora čine ulazne značajke za naše modele.

Meteorološke varijable

Uključeni su okolišni uvjeti koji utječu i na razine onečišćenja i na ponašanje senzora:

- Temperatura
- Relativna vlažnost
- Apsolutna vlažnost

Napomene o vremensko-serijskoj prirodi podataka

Sva su mjerenja vremenski označena i prikupljena sekvencijalno. To znači da su susjedna opažanja vremenski povezana i nisu potpuno neovisna.

Za ovo predavanje, međutim, svako se mjerenje tretira kao snimka okoliša u određenom satu. Kako bismo izbjegli metodološke probleme, podatke dijelimo kronološki na treniranje i testiranje, oponašajući način na koji bi se takvi sustavi koristili u praksi.

Napomene o izvornoj definiciji problema

Prema UCI repozitoriju, ovaj je skup podataka namijenjen regresiji. Cilj izvornog problema bio je predvidjeti kontinuirane koncentracije onečišćivala (kao što je NO_2) iz signala senzora i meteoroloških podataka.

Takva formulacija ima smisla kada je cilj precizno numeričko predviđanje.

Preformuliranje problema za klasifikaciju

U praćenju okoliša točne vrijednosti onečišćivala često su manje važne od toga je li prekoračen kritični prag. Regulatorni sustavi i javna upozorenja obično se aktiviraju kada onečišćenje prijeđe unaprijed definirane granice.

Zbog toga izvorni regresijski problem preformuliramo u zadatak binarne klasifikacije. Umjesto predviđanja točne koncentracije NO_2, predviđamo događa li se prekoračenje.

Korišteni regulatorni prag

Prag korišten u ovoj studiji slučaja potječe iz službenog europskog zakonodavstva o kvaliteti zraka:

Direktiva 2008/50/EZ Europskog parlamenta i Vijeća od 21. svibnja 2008. o kvaliteti okolnog zraka i čišćem zraku za Europu. Iz te direktive proizlazi:

- Granična vrijednost NO_2 po satu iznosi 200 µg/m³
- Ta se granica ne smije prekoračiti više od 18 puta godišnje.

Službeni zakonodavni tekst (EUR-Lex) za ovu direktivu navodi granice u PRILOGU XI. (Granične vrijednosti za zaštitu zdravlja ljudi), u odjeljku B.

Korištenje tog praga osigurava da je naš klasifikacijski zadatak pravno utemeljen i okolišno smislen.

Definicija dviju klasa

Na temelju propisa EU-a ciljnu varijablu definiramo kako slijedi:

- **Klasa 0 – prihvatljiva kvaliteta zraka: koncentracija NO_2 manja od 200 µg/m³**
- **Klasa 1 – događaj prekoračenja: koncentracija NO_2 jednaka 200 µg/m³ ili veća**

Time se stvaraju dvije jasne i interpretabilne klase.

Zašto je logistička regresija prikladna za ovo uvodno predavanje

U ovoj se preformulaciji logistička regresija koristi za procjenu vjerojatnosti da će se dogoditi prekoračenje NO_2 s obzirom na dostupne senzorske i meteorološke podatke.

Logistička regresija dobro je prikladna za ovaj zadatak jer:

- Daje procjene vjerojatnosti, a ne samo oznake klasa,
- njezine su klasifikacijske granice lako interpretabilne,

- omogućuje jasnu demonstraciju inženjeringa značajki,
- omogućuje jednostavno uočavanje prenaučenosti i podnaučenosti,
- podržava intuitivnu vizualizaciju ponašanja modela.

Priroda vremensko-serijskog skupa podataka u kontekstu preformulacije problema

Iako se podaci prikupljaju kroz vrijeme, to ne predstavlja problem za prikazani slučaj uporabe.

U ovom predavanju:

- Ne modeliramo eksplicitno vremensku dinamiku,
- svako se opažanje tretira kao neovisna snimka,
- kronološka podjela na treniranje i testiranje koristi se kako bi se izbjeglo curenje informacija.

Tehnike modeliranja vremenskih nizova namjerno su izostavljene iz ove jedinice kako bi se zadržao fokus na temeljnim konceptima strojnog učenja.

Stoga prilagođeni slučaj uporabe omogućuje istraživanje širokog raspona važnih koncepata strojnog učenja u realističnom okolišnom kontekstu:

- pretvaranje regresijskog skupa podataka u klasifikacijski zadatak pomoću regularnih direktiva zakonodavnih tijela,
- inženjering značajki na temelju okolišnog znanja,
- razumijevanje linearnih i nelinearnih granica odlučivanja,
- ispravno odvajanje podataka za treniranje i testiranje,
- prepoznavanje prenaučenosti i podnaučenosti,
- korištenje vizualizacije za interpretaciju i analizu modela.

Kao rezultat toga, slučaj uporabe pruža jasnu i praktičnu vezu između teorije strojnog učenja i stvarnih okolišnih primjena .

S obzirom na ograničenja iz direktive, fokusiramo se na klasifikaciju prekoračenja po satu. Najprije ćemo učitati podatke pomoću paketa ucimlrepo. Ostale biblioteke i paketi potrebni u ovom izlaganju jesu scikit-learn , pandas, numpy i matplotlib.

[1] ✓ 8 s

```
# If running in Google Colab, install dependencies (safe to run even if already installed)
!pip -q install ucimlrepo scikit-learn pandas numpy matplotlib
```

[2] ✓ 5 s

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, ConfusionMatrixDisplay, roc_auc_score, RocCurveDisplay
)

# For reproducibility
RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)


from sklearn.dummy import DummyClassifier
from sklearn.metrics import (
    precision_recall_curve, average_precision_score, PrecisionRecallDisplay
)

```

Model je LogisticRegression iz scikit-learna, a metrike su standardne mjere za evaluaciju uspješnosti modela. Random state osigurava ~~reproduktivnost~~reproducibilnost rezultata. DummyClassifier je jednostavan klasifikator koji svim instancama dodjeljuje većinsku klasu (najčešću klasu) – potreban nam je kako bismo pokazali probleme s nekim standardnim metrikama u tzv. problemima neuravnotežene klasifikacije (ovaj slučaj uporabe pripada tom tipu). Učitavanje podataka izvodi se pomoću ucimlrepo bez potrebe za izravnim preuzimanjem.

[3] ✓ 1 s

```
from ucimlrepo import fetch_ucirepo

# UCI ML Repo dataset: Air Quality
# Note: In ucimlrepo, dataset IDs can change or multiple IDs can exist.
# We attempt fetch by name; if it fails, we'll guide a fallback.
try:
    air_quality = fetch_ucirepo(name="Air Quality")
    X_raw = air_quality.data.features
    y_raw = air_quality.data.targets  # may be empty for some UCI loaders
    meta = air_quality.metadata
    print("Loaded from UCI via ucimlrepo.")
    print("Features shape:", X_raw.shape)
    print("Targets shape:", None if y_raw is None else y_raw.shape)
    display(meta)
except Exception as e:
    print("Could not fetch via ucimlrepo:", repr(e))
    print("\nFallback option: download the dataset from UCI and upload to Colab, then read it with pandas.")
    raise

```

Čišćenje nedostajućih vrijednosti i parsiranje vremenskih oznaka dio su postupka pripreme podataka. U tom smislu ćemo:

- Kombinirati Date i Time u vremensku oznaku
- zamijeniti -200 s NaN
- izbaciti retke u kojima nedostaje temeljna istina za NO_2 (potrebna nam je za oznake)
- (po potrebi) izbaciti stupce s previše nedostajućih vrijednosti

[4]
✓ 0 s

```python
df = X_raw.copy()

# In many versions of the dataset, Date and Time columns exist like 'Date' and 'Time'
# We'll try to detect them robustly.
date_col_candidates = [c for c in df.columns if c.lower() in ("date", "data")]
time_col_candidates = [c for c in df.columns if c.lower() in ("time", "ora")]

print("Date candidates:", date_col_candidates)
print("Time candidates:", time_col_candidates)

if not date_col_candidates or not time_col_candidates:
    raise ValueError("Could not find Date/Time columns. Inspect df.columns and adjust parsing.")

date_col = date_col_candidates[0]
time_col = time_col_candidates[0]

# Create a timestamp (common format is dd/mm/yyyy and hh.mm.ss or hh.mm)
# We'll try flexible parsing.
df["timestamp"] = pd.to_datetime(df[date_col].astype(str) + " " + df[time_col].astype(str),
                                 errors="coerce", dayfirst=True)

# Replace -200 with NaN in numeric columns
for c in df.columns:
    if c == "timestamp":
        continue
    # Attempt numeric conversion if possible
    df[c] = pd.to_numeric(df[c], errors="ignore")

num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols] = df[num_cols].replace(-200, np.nan)

# Sort by time
df = df.sort_values("timestamp").reset_index(drop=True)

# Identify the NO2 ground-truth column (commonly 'NO2(GT)')
no2_candidates = [c for c in df.columns if "no2" in c.lower() and "gt" in c.lower()]
print("NO2(GT) candidates:", no2_candidates)
if not no2_candidates:
    raise ValueError("Could not find NO2(GT) column. Inspect df.columns to locate it.")
NO2_COL = no2_candidates[0]

# Drop rows without timestamp or without NO2 ground truth
df = df.dropna(subset=["timestamp", NO2_COL]).reset_index(drop=True)

print("After cleaning:")
print(df.shape)
df.head()

```

Podatkovni okvir nakon čišćenja:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH	timestamp
0	4/1/2004	0:00:00	1.6	1143.0	106.0	6.3	825.0	96.0	986.0	86.0	1477.0	978.0	12.0	61.6	0.8593	2004-01-04 00:00:00
1	4/1/2004	1:00:00	1.2	1044.0	100.0	5.1	770.0	85.0	1031.0	70.0	1425.0	944.0	11.5	63.9	0.8652	2004-01-04 01:00:00
2	4/1/2004	2:00:00	1.1	1034.0	71.0	4.1	716.0	50.0	1085.0	55.0	1405.0	891.0	10.7	67.2	0.8630	2004-01-04 02:00:00
3	4/1/2004	4:00:00	0.7	909.0	44.0	2.4	615.0	57.0	1237.0	49.0	1322.0	790.0	10.2	66.6	0.8299	2004-01-04 04:00:00
4	4/1/2004	5:00:00	0.9	996.0	45.0	2.9	648.0	64.0	1176.0	50.0	1340.0	852.0	11.0	63.7	0.8325	2004-01-04 05:00:00

Kako bismo izvorni regresijski problem preformulirali u klasifikacijski problem, koristit ćemo razine prekoračenja EU-a iz navedene direktive. Izrađujemo oznake:

- **y = 1 ako je $NO_2 \geq 200$ µg/m³**
- **y = 0 u suprotnom**

To odgovara stvarnom ~~političko-regulatornom~~regulatornom pragu, pa klase imaju značenje u stvarnom svijetu.

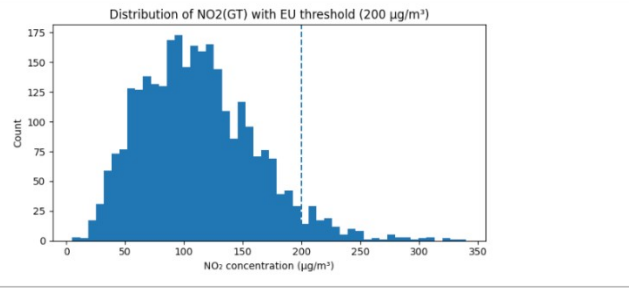
```
[6]
✓ 0 s
1 df["y_exceed"] = (df[NO2_COL] >= EU_NO2_THRESHOLD).astype(int)
2
3 pos_rate = df["y_exceed"].mean()
4 print(f"Positive class rate (NO₂ exceedance): {pos_rate:.4f} ({pos_rate*100:.2f}%)")
5 df[["timestamp", NO2_COL, "y_exceed"]].head()
6
```

```
Positive class rate (NO₂ exceedance): 0.0478 (4.78%)
```

	timestamp	NO2(GT)	y_exceed
0	2004-01-04 00:00:00	86.0	0
1	2004-01-04 01:00:00	70.0	0
2	2004-01-04 02:00:00	55.0	0
3	2004-01-04 04:00:00	49.0	0
4	2004-01-04 05:00:00	50.0	0

Iz gore navedenih podataka vidljivo je da samo 4,78 % ukupnih podatkovnih točaka pripada klasi 1 (razina prekoračenja), što je tipičan primjer neuravnotežene klasifikacije. Iz vizualizacije distribucije NO2 u odnosu na prag EU-a neuravnoteženost se vidi još jasnije.

```
[22]
✓ 0 s
1 # NO2 distribution with EU threshold line
2 EU_NO2_THRESHOLD = 200.0  # µg/m³
3
4 plt.figure(figsize=(7,4))
5 plt.hist(df[NO2_COL].dropna(), bins=50)
6 plt.axvline(EU_NO2_THRESHOLD, linestyle="--")
7 plt.title(f"Distribution of {NO2_COL} with EU threshold (200 µg/m³)")
8 plt.xlabel("NO₂ concentration (µg/m³)")
9 plt.ylabel("Count")
10 plt.tight_layout()
11 plt.show()
12
13 # Class balance preview (will define label in next section)
14
15
```



Nastavljamo s pripremom podataka (imputacija medijanom) i inženjeringom značajki (odabir značajki) tako da izbacujemo stupce koji nisu značajke, Date, Time i timestamps, zadržavamo samo numeričke senzorske i meteorološke varijable te naposljetku izbacujemo rijetke stupce.

[8]
✓ 0s

```
# Drop non-features
drop_cols = [date_col, time_col, "timestamp", NO2_COL, "y_exceed"]
feature_df = df.drop(columns=[c for c in drop_cols if c in df.columns], errors="ignore")

# Keep numeric features only
X_all = feature_df.select_dtypes(include=[np.number]).copy()
y_all = df["y_exceed"].copy()

# Optionally drop columns with too much missing data (e.g., > 40% missing)
too_missing = X_all.columns[X_all.isna().mean() > 0.40]
print("Dropping columns with >40% missing:", list(too_missing))
X_all = X_all.drop(columns=too_missing)

# Median imputation
X_all = X_all.fillna(X_all.median(numeric_only=True))

print("Final feature matrix:", X_all.shape)
X_all.head()
```

```
Dropping columns with >40% missing: ['NMHC(GT)']
Final feature matrix: (2910, 11)
```

	CO(GT)	PT08.S1(CO)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
0	1.6	1143.0	6.3	825.0	96.0	986.0	1477.0	978.0	12.0	61.6	0.8593
1	1.2	1044.0	5.1	770.0	85.0	1031.0	1425.0	944.0	11.5	63.9	0.8652
2	1.1	1034.0	4.1	716.0	50.0	1085.0	1405.0	891.0	10.7	67.2	0.8630
3	0.7	909.0	2.4	615.0	57.0	1237.0	1322.0	790.0	10.2	66.6	0.8299
4	0.9	996.0	2.9	648.0	64.0	1176.0	1340.0	852.0	11.0	63.7	0.8325

Sada možemo pokazati zašto apsolutna točnost može biti zavaravajuća u izrazito neuravnoteženim skupovima podataka. Uz prag EU-a (NO_2 ≥ 200 µg/m³), pozitivna je klasa obično rijetka (problem detekcije događaja). To znači da sama točnost može zavarati: model koji uvijek predviđa „nema prekoračenja" može postići visoku točnost, a ipak biti beskoristan. Računamo osnovnu vrijednost za većinsku klasu (pomoću DummyClassifiera) i uvodimo preciznost/odziv te PR-AUC (Average Precision) kao informativnije metrike za rijetke događaje. Izlaz dobiven iz ovog ilustrativnog primjera upravo je ono što želimo za demonstraciju „zamke točnosti" — savršena ilustracija neuravnotežene klasifikacije.

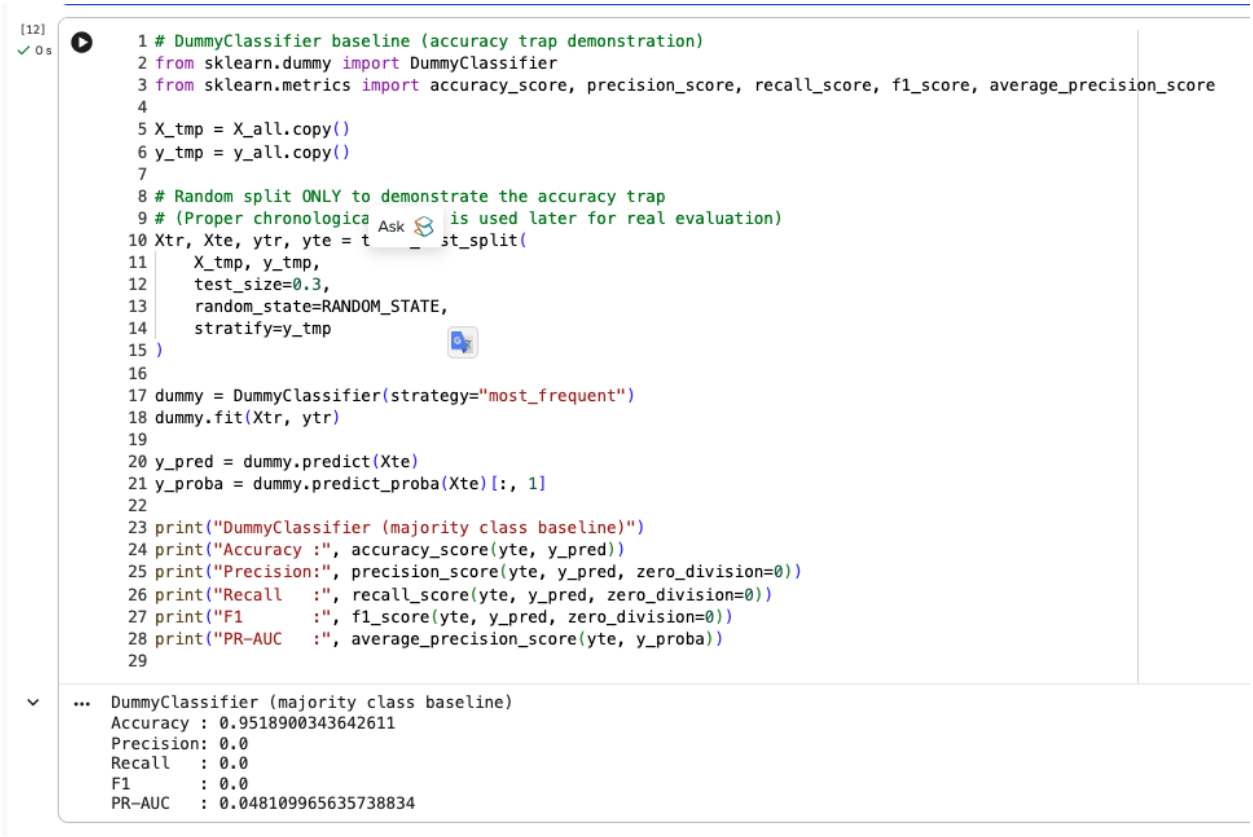
- Točnost ≈ 0,952 → Oko 95,2 % sati u našoj podjeli pripada klasi „nema prekoračenja". Dummy model predviđa samo većinsku klasu (0), pa prema točnosti izgleda „dobro".

- Preciznost = 0,0, odziv = 0,0, F1 = 0,0 → Model nikada ne predviđa prekoračenje (klasa 1). Dakle:

    - Hvata 0 prekoračenja → odziv = 0

- Daje 0 pozitivnih predviđanja → preciznost je definirana kao 0 (uz zero_division = 0)
- F1 je stoga 0

- PR-AUC ≈ 0,0481 To je ovdje najinformativniji broj. Za takav dummy model (koji uvijek predviđa isto), PR-AUC teži biti blizu udjela pozitivne klase. Stoga je naša stopa prekoračenja vjerojatno oko 4,8 % (barem u toj evaluacijskoj podjeli). To je u skladu s točnošću od 95 %.



```
[12]
✓ 0s
```

```python
# DummyClassifier baseline (accuracy trap demonstration)
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, average_precision_score

X_tmp = X_all.copy()
y_tmp = y_all.copy()

# Random split ONLY to demonstrate the accuracy trap
# (Proper chronologica Ask is used later for real evaluation)
Xtr, Xte, ytr, yte = t st_split(
    X_tmp, y_tmp,
    test_size=0.3,
    random_state=RANDOM_STATE,
    stratify=y_tmp
)

dummy = DummyClassifier(strategy="most_frequent")
dummy.fit(Xtr, ytr)

y_pred = dummy.predict(Xte)
y_proba = dummy.predict_proba(Xte)[:, 1]

print("DummyClassifier (majority class baseline)")
print("Accuracy :", accuracy_score(yte, y_pred))
print("Precision:", precision_score(yte, y_pred, zero_division=0))
print("Recall   :", recall_score(yte, y_pred, zero_division=0))
print("F1       :", f1_score(yte, y_pred, zero_division=0))
print("PR-AUC   :", average_precision_score(yte, y_proba))
```

```
DummyClassifier (majority class baseline)
Accuracy : 0.9518900343642611
Precision: 0.0
Recall   : 0.0
F1       : 0.0
PR-AUC   : 0.048109965635738834
```

Naš model logističke regresije treba učiti iz podataka, ali također zahtijevamo da dobro generalizira na neviđene podatke iz iste problemske domene. Zbog toga se skup podataka redovito dijeli na skup za treniranje i testni skup. Budući da je izvorni skup podataka vremenski niz, kako bismo izbjegli curenje podataka, koristit ćemo prvih 70 % podataka kao skup za treniranje, a posljednjih 30 % kao testni skup (kronološka podjela). Time se simulira realistična situacija: treniranje na prošlosti, evaluacija na budućnosti. Napomena o neuravnoteženosti: nakon pragiranja (granična vrijednost EU-a za NO_2), događaji prekoračenja mogu biti rijetki. Stoga se oslanjamo na preciznost/odziv, PR-AUC i matrice zabune — ne samo na točnost. Općenito želimo da model dobro radi i na skupu

za treniranje i na testnom skupu. Za vizualizaciju granica odlučivanja odabiremo dvije značajke (mogu se koristiti bilo koje dvije informativne značajke).

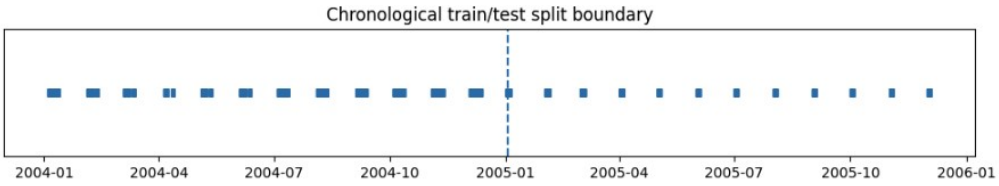
Zatim treniramo logističku regresiju sa standardizacijom i crtamo:

- raspršeni prikaz točaka (obojenih prema klasi)
- linearnu granicu odlučivanja
- konture vjerojatnosti (neobvezno)

[13]
✓ 0 s

```
# Chronological split index
split_idx = int(len(df) * 0.70)

X_train = X_all.iloc[:split_idx].copy()
y_train = y_all.iloc[:split_idx].copy()
X_test  = X_all.iloc[split_idx:].copy()
y_test  = y_all.iloc[split_idx:].copy()

print("Train size:", X_train.shape, "Test size:", X_test.shape)
print("Train positive rate:", y_train.mean(), "Test positive rate:", y_test.mean())

# Visualize split on timeline
Ask ure(figsize=(10,2))
plt.plot(df["timestamp"], np.zeros(len(df)), marker="|", linestyle="None")
plt.axvline(df["timestamp"].iloc[split_idx], linestyle="--")
plt.title("Chronological train/test split boundary")
plt.yticks([])
plt.tight_layout()
plt.show()

```

```
Train size: (2036, 11) Test size: (874, 11)
Train positive rate: 0.007858546168958742 Test positive rate: 0.14073226544622425
```



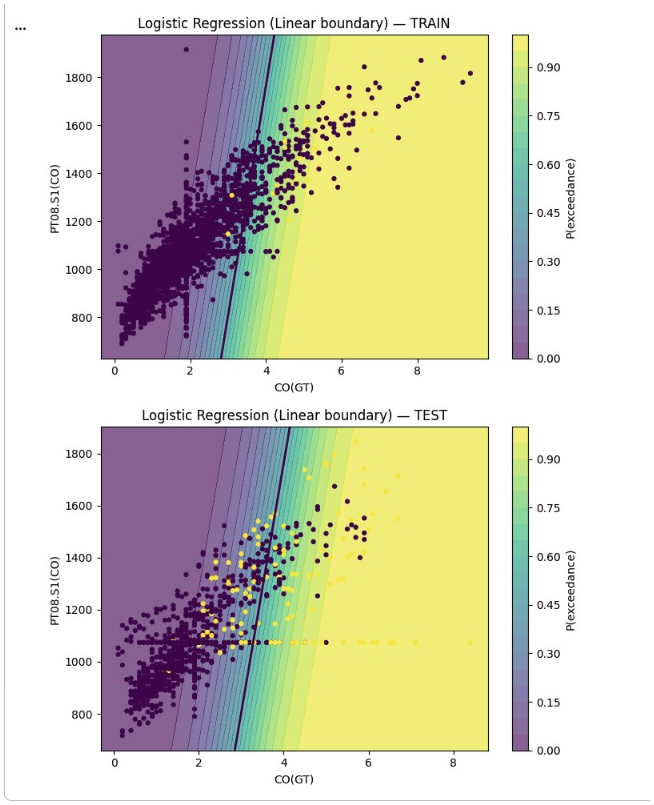
[14]
✓ 0s

```python
# Pick two features to visualize
# We'll choose two columns with decent variance
candidate_cols = X_all.columns.tolist()
if len(candidate_cols) < 2:
    raise ValueError("Not enough numeric features to visualize.")

feat1, feat2 = candidate_cols[0], candidate_cols[1]
print("Using features:", feat1, "and", feat2)

X2_train = X_train[[feat1, feat2]].copy()
X2_test  = X_test[[feat1, feat2]].copy()

lin_model = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(max_iter=2000, class_weight="balanced", random_state=RANDOM_STATE))
])
lin_model.fit(X2_train, y_train)

print("Train accuracy:", accuracy_score(y_train, lin_model.predict(X2_train)))
print("Test accuracy :", accuracy_score(y_test,  lin_model.predict(X2_test)))

# More informative metrics on TEST (imbalanced setting)
y_pred_lin = lin_model.predict(X2_test)
y_proba_lin = lin_model.predict_proba(X2_test)[:,1]

print("TEST Precision:", precision_score(y_test, y_pred_lin, zero_division=0))
print("TEST Recall   :", recall_score(y_test, y_pred_lin, zero_division=0))
print("TEST F1       :", f1_score(y_test, y_pred_lin, zero_division=0))
if len(np.unique(y_test)) == 2:
    print("TEST PR-AUC (AP):", average_precision_score(y_test, y_proba_lin))

```

```
Using features: CO(GT) and PT08.S1(CO)
Train accuracy: 0.8742632612966601
Test accuracy : 0.8649885583524027
TEST Precision: 0.5177304964539007
TEST Recall   : 0.5934959349593496
TEST F1       : 0.553030303030303
TEST PR-AUC (AP): 0.593996588340624
```

[15] ✓ 0 s

```python
def plot_decision_boundary_2d(model, X, y, title, grid_steps=200):
    # Create grid
    x_min, x_max = X.iloc[:,0].min(), X.iloc[:,0].max()
    y_min, y_max = X.iloc[:,1].min(), X.iloc[:,1].max()
    pad_x = 0.05 * (x_max - x_min + 1e-9)
    pad_y = 0.05 * (y_max - y_min + 1e-9)
    x_min, x_max = x_min - pad_x, x_max + pad_x
    y_min, y_max = y_min - pad_y, y_max + pad_y

    xx, yy = np.meshgrid(np.linspace(x_min, x_max, grid_steps),
                         np.linspace(y_min, y_max, grid_steps))
    grid = pd.DataFrame({X.columns[0]: xx.ravel(), X.columns[1]: yy.ravel()})

    # Predict probabilities for class 1
    probs = model.predict_proba(grid)[:,1].reshape(xx.shape)

    plt.figure(figsize=(7,5))
    # Probability contour
    cs = plt.contourf(xx, yy, probs, levels=20, alpha=0.6)
    plt.colorbar(cs, label="P(exceedance)")

    # Scatter points
    plt.scatter(X.iloc[:,0], X.iloc[:,1], c=y, s=12)
    # Decision boundary at p=0.5
    plt.contour(xx, yy, probs, levels=[0.5], linewidths=2)

    plt.xlabel(X.columns[0])
    plt.ylabel(X.columns[1])
    plt.title(title)
    plt.tight_layout()
    plt.show()

plot_decision_boundary_2d(lin_model, X2_train, y_train,
                          "Logistic Regression (Linear boundary) — TRAIN")
plot_decision_boundary_2d(lin_model, X2_test, y_test,
                          "Logistic Regression (Linear boundary) — TEST")
```



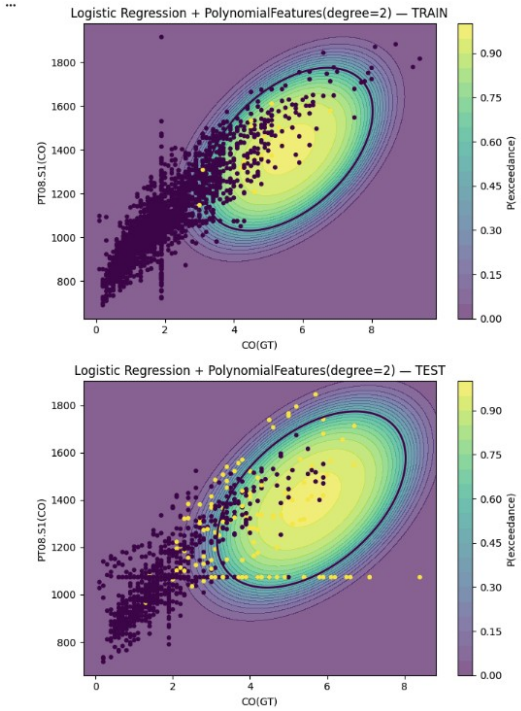
Vizualizacija linearne granice odlučivanja pokazuje kako logistička regresija razdvaja dvije klase ravnom linijom pri pragu vjerojatnosti (p = 0,5), otkrivajući i razdvajanje klasa i pouzdanost modela u različitim područjima prostora značajki.

Logistička regresija linearna je u svojim parametrima, ali transformacijom ulaznih značajki (npr. dodavanjem kvadratnih i interakcijskih članova) možemo dobiti nelinearnu granicu odlučivanja u izvornom prostoru. Koristimo PolynomialFeatures (degree=2) za provedbu dodatnog inženjeringa značajki.

```
[23]
✓ 4 s
```

```python
poly2_model = Pipeline([
    ("scaler", StandardScaler()),
    ("poly", PolynomialFeatures(degree=2, include_bias=False)),
    ("clf", LogisticRegression(max_iter=5000, class_weight="balanced", random_state=RANDOM_STATE))
])
poly2_model.fit(X2_train, y_train)

print("Train accuracy (degree=2):", accuracy_score(y_train, poly2_model.predict(X2_train)))
print("Test accuracy  (degree=2):", accuracy_score(y_test,  poly2_model.predict(X2_test)))
# More informative metrics on TEST
y_pred_poly2 = poly2_model.predict(X2_test)
y_proba_poly2 = poly2_model.predict_proba(X2_test)[:,1]
print("TEST Precision (deg=2):", precision_score(y_test, y_pred_poly2, zero_division=0))
print("TEST Recall    (deg=2):", recall_score(y_test, y_pred_poly2, zero_division=0))
print("TEST F1        (deg=2):", f1_score(y_test, y_pred_poly2, zero_division=0))
if len(np.unique(y_test)) == 2:
    print("TEST PR-AUC (AP) (deg=2):", average_precision_score(y_test, y_proba_poly2))


plot_decision_boundary_2d(poly2_model, X2_train, y_train,
                          "Logistic Regression + PolynomialFeatures(degree=2) — TRAIN")
plot_decision_boundary_2d(poly2_model, X2_test, y_test,
                          "Logistic Regression + PolynomialFeatures(degree=2) — TEST")


```

```
Train accuracy (degree=2): 0.8487229862475442
Test accuracy  (degree=2): 0.8260869565217391
TEST Precision (deg=2): 0.4013605442176871
TEST Recall    (deg=2): 0.4796747967479675
TEST F1        (deg=2): 0.4370370703703706
TEST PR-AUC (AP) (deg=2): 0.44004674954757994
```



Vizualizacija nelinearne granice odlučivanja prikazuje zakrivljeno razdvajanje nastalo inženjeringom značajki, što povećava fleksibilnost modela, ali i rizik od prenaučenosti u šumnim, neuravnoteženim okolišnim podacima. Usporedbom uspješnosti modela na oba skupa podataka linearna logistička regresija nadmašuje nelinearni model (degree = 2), što upućuje na to da je jednostavna linearna granica dovoljna, dok nelinearni model postaje pretjerano složen i počinje se prilagođavati šumu u podacima.

Ključan pojam u strojnom učenju jest prenaučenost i podnaučenost modela. U kontroliranom eksperimentu objasnit ćemo te fenomene kontroliranjem složenosti modela pomoću stupnja polinoma (složenosti značajki) i jačine regularizacije C u logističkoj regresiji. Konkretno, za demonstraciju podnaučenosti isključit ćemo težinjenje klasa u logističkoj regresiji. Promatranjem uspješnosti na skupu za treniranje i testnom skupu moguće je uočiti nijanse tih problema.

[22]
✓ 1s

```python
# Use TWO features (still underfitting due to imbalance + no class weighting)
X2u_train = X_train[[feat1, feat2]].copy()
X2u_test  = X_test[[feat1, feat2]].copy()

underfit_model = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(
        max_iter=4000,
        C=1.0,                  # normal regularization; underfitting comes from no class weighting
        class_weight=None,
        random_state=RANDOM_STATE
    ))
])

underfit_model.fit(X2u_train, y_train)
y_pred_tr = underfit_model.predict(X2u_train)
y_proba_tr = underfit_model.predict_proba(X2u_train)[:, 1]
y_pred_te = underfit_model.predict(X2u_test)
y_proba_te = underfit_model.predict_proba(X2u_test)[:, 1]

metrics_report(y_train, y_pred_tr, y_proba_tr, "UNDERFITTING model — TRAIN (2 features, no class weight)")
metrics_report(y_test,  y_pred_te, y_proba_te, "UNDERFITTING model — TEST  (2 features, no class weight)")

# Visualization in 2D: decision boundary + probability surface
plot_decision_boundary_2d(underfit_model, X2u_train, y_train,
                          "Underfitting example: 2D decision boundary — TRAIN (no class weights)")
plot_decision_boundary_2d(underfit_model, X2u_test, y_test,
                          "Underfitting example: 2D decision boundary — TEST (no class weights)")

```

```
UNDERFITTING model — TRAIN (2 features, no class weight)
Accuracy : 0.992
Precision: 0.000
Recall   : 0.000
F1       : 0.000
PR-AUC(AP): 0.070

UNDERFITTING model — TEST  (2 features, no class weight)
Accuracy : 0.859
Precision: 0.000
Recall   : 0.000
F1       : 0.000
PR-AUC(AP): 0.591
```

Ovaj primjer podnaučenosti dobiven je namjernim uklanjanjem težinjenja klasa . Bez te korekcije model logističke regresije ponajprije uči predviđati većinsku klasu i potpuno propušta događaje prekoračenja NO_2. Visoka točnost opažena na testnom skupu i skupu za treniranje posljedica je neuravnoteženosti klasa, dok su odziv, F1-mjera i PR-AUC niski na oba skupa podataka, što pokazuje da model ne uspijeva detektirati rijetke, ali važne slučajeve. To naglašava kako model može izgledati uspješno ako se procjenjuje samo prema točnosti, čak i kada ne uči smisleno ili praktično pravilo odlučivanja.

```python
overfit_degree = 7
overfit_C = 50.0

overfit_model = Pipeline([
    ("poly", PolynomialFeatures(degree=overfit_degree, include_bias=False)),
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(
        solver="lbfgs",
        max_iter=50000,
        C=overfit_C,
        class_weight="balanced",
        random_state=RANDOM_STATE
    ))
])
overfit_model.fit(X2_train, y_train)

y_pred_tr = overfit_model.predict(X2_train)
y_proba_tr = overfit_model.predict_proba(X2_train)[:,1]
y_pred_te = overfit_model.predict(X2_test)
y_proba_te = overfit_model.predict_proba(X2_test)[:,1]

metrics_report(y_train, y_pred_tr, y_proba_tr, f"OVERFITTING model — TRAIN (deg={overfit_degree}, C={overfit_C})")
metrics_report(y_test,  y_pred_te, y_proba_te, f"OVERFITTING model — TEST  (deg={overfit_degree}, C={overfit_C})")

plot_decision_boundary_2d(overfit_model, X2_train, y_train, f"Overfitting: nonlinear boundary (deg={overfit_degree}) — TRAIN")
plot_decision_boundary_2d(overfit_model, X2_test,  y_test,  f"Overfitting: nonlinear boundary (deg={overfit_degree}) — TEST")
```

```
OVERFITTING model — TRAIN (deg=7, C=50.0)
Accuracy : 0.854
Precision: 0.045
Recall   : 0.875
F1       : 0.086
PR-AUC(AP): 0.128

OVERFITTING model — TEST  (deg=7, C=50.0)
Accuracy : 0.838
Precision: 0.437
Recall   : 0.537
F1       : 0.482
PR-AUC(AP): 0.520
```

U slučaju prenaučenosti model postaje pretjerano fleksibilan i previše se prilagođava podacima za treniranje. Kao rezultat toga detektira većinu događaja prekoračenja, ali proizvodi mnogo lažnih alarma, što dovodi do niske preciznosti i F1-mjere na skupu za treniranje. Kada se isti model primijeni na nove podatke, takvo se ponašanje ne prenosi: odziv se smanjuje, preciznost se mijenja, a ukupna uspješnost postaje manje pouzdana. Jasna razlika između rezultata na skupu za treniranje i testnom skupu upućuje na to da je model naučio obrasce koji se ne generaliziraju dobro, što je tipičan znak prenaučenosti.

Praktična vježba: detekcija staništa smreke/jele pomoću logističke regresije

Skup podataka za ovu vježbu: koristimo skup podataka Forest Cover Type (često nazivan Covertype). Sadrži kartografske i okolišne varijable (nadmorska visina, nagib, udaljenosti do hidroloških objekata, zasjenjenje terena, područja divljine, tipovi tla, …), a oznaka je tip šumskog pokrova.

U okolišnom smislu, ovo je tipičan zadatak:

- **Predvidjeti klase staništa/pokrova iz terenskih i okolišnih deskriptora.**

Izvorni skup podataka obuhvaća 7 tipova pokrova (višeklasni problem). Kako bi vježba ostala usklađena s binarnom logističkom regresijom i kako bi se metrike/grafovi lakše interpretirali, pretvaramo ga u binarni zadatak:

- **pozitivna klasa (1): Cover Type 1 = Spruce/Fir**
- **negativna klasa (0): svi ostali tipovi pokrova**

Ova preformulacija odražava realističan okolišni scenarij: detekciju specifičnog tipa staništa od interesa nasuprot svim ostalim tipovima zemljišnog pokrova. Skup podataka može se dohvatiti izravno pomoću scikit-learnove funkcije fetch_covtype().

Alati i okruženje

- Programski jezik: Python (Google Colab)
- Biblioteke: numpy, pandas, scikit-learn, matplotlib
- Pristup skupu podataka: sklearn.datasets.fetch_covtype
- Način rada: individualni rad studenta

Zadatak opisa značajki

Studenti moraju konzultirati izvornu dokumentaciju skupa podataka i izraditi tablicu opisa značajki koja sadrži sljedeće informacije:

- Naziv značajke
- Tip značajke (kontinuirana ili kategorijska / kodirana metodom one-hot)
- Jedinice (ako je primjenjivo)
- Fizičko ili okolišno značenje

Skup podataka obuhvaća 10 kontinuiranih okolišnih varijabli i 44 kategorijske varijable (4 indikatora područja divljine i 40 indikatora tipova tla, kodiranih metodom one-hot).

Zadaci studenata (korak po korak)

1. Učitajte podatke iz scikit-learna pomoću fetch_covtype()
2. Skup podataka je velik, stoga uzorkujte podskup (N=50000 zapisa) koristeći stratificirano uzorkovanje

[3.] Uvezite sve biblioteke (isto kao u slučaju uporabe iz predavanja) i osigurajte ~~reproduktivnost~~reproducibilnost eksperimenta postavljanjem fiksnog slučajnog sjemena

3.[4.] Pretvorite višeklasni problem u problem binarne klasifikacije

a. Klasa označena kao 1 = Spruce/Fir (pozitivna klasa od interesa)
b. Klasa označena kao 0 = svi ostali tipovi (negativna klasa)

Tretirajte Spruce/Fir kao pozitivnu klasu jer je glavni cilj detektirati stanište od interesa!

4.[5.] Ispitajte i izvijestite o ravnoteži klasa između pozitivne i negativne klase

5.[6.] Objasnite zašto je binarna preformulacija realistična u praćenju okoliša

6.[7.] Provjerite nedostajuće podatke

a. Ako nedostaje malo vrijednosti → izbacite ih
b. Ako ih je mnogo → imputirajte (sredinom ili modom, ovisno o tipu značajke)
c. Ako ih nema, izričito navedite „nije potrebna nikakva radnja."

7.[8.] Provjerite duplikate (neobvezno) i osnovne izdvojenice (barem za kontinuirane varijable)

8.[9.] Izradite skup za treniranje i testni skup (70/30), uz zadržavanje iste distribucije klasa pomoću stratify = y, i objasnite ulogu svake podjele

9.[10.] Pogledajte skale prvih 10 značajki i zaključite što je potrebno za logističku regresiju kao početni korak pripreme podataka; važno: na kojem se skupu podataka to mora učiniti?→

a. Koristite Pipeline sa StandardScalerom.
b. Covertype ima mnogo 0/1 one-hot značajki; sigurnije je koristiti StandardScaler(with_mean=False).
c. Objasnite zašto je skaliranje važno za LR (regularizacija ovisi o skali značajki)

10.[11.] Trenirajte osnovni model logističke regresije; uzmite u obzir da regularizacija kontrolira složenost: manji C = jača regularizacija → jednostavniji model (može biti podnaučen); veći C = slabija regularizacija → fleksibilniji model (može biti prenaučen, osobito s inženjeringom značajki). Logistička regresija daje vjerojatnosti: zadani prag od 0,5 proizvoljan je.→

11.[12.] Izvijestite evaluacijske metrike na skupu za treniranje i testnom skupu (koje ste metrike odabrali i zašto)

12.[13.] Prikažite matricu zabune na testnom skupu i objasnite:

- Objasnite što TP, TN, FP i FN znače u ovom okolišnom kontekstu.
- FP → model predviđa „Spruce/Fir", ali je lokacija neki drugi tip pokrova; okolišno rizično: pogrešna karta staništa → pogrešne upravljačke/konzervacijske odluke→→
- FN → model predviđa „Other", ali je lokacija zapravo „Spruce/Fir" → okolišno skupo, ali manje opasno od FP: stanište je nedovoljno zaštićeno/podcijenjeno→
- To pokazuje kako ML podupire okolišno odlučivanje.

13.[14.] Usporedite metrike iz koraka 9 i interpretirajte kompromise između evaluacijskih mjera.

a. Objasnite o kojoj je vrsti problema riječ i zašto ga je važno detektirati; razmotrite detekciju FP i FN u odnosu na trošak propuštanja staništa nasuprot lažnom alarmu.
b. Ovo je problem detekcije staništa s asimetričnim troškovima, pa promjena praga utječe na FP/FN.

14.[15.] Objasnite preciznost, odziv i PR-AUC posebno za zadatak detekcije staništa Spruce/Fir.

15.[16.] Povećajte fleksibilnost modela pomoću polinomskog proširenja značajki (stupnjevi 1, 2 i 4) primijenjenog samo na kontinuirane značajke te mijenjajte jačinu regularizacije (C = 0,1, 1,0, 50). Objasnite očekivane učinke na ponašanje modela.

a. Kako biste izbjegli rušenje Colaba, primijenite polinomske značajke samo na 10 kontinuiranih varijabli.

16.[17.] Vizualizirajte F1-mjeru na skupovima za treniranje i testiranje za sve konfiguracije modela i interpretirajte opažene obrasce.

17.[18.] Objasnite koncepte podnaučenosti i prenaučenosti koristeći rezultate iz prethodnog koraka.

18.[19.] Identificirajte jednu konfiguraciju podnaučenosti i jednu konfiguraciju prenaučenosti te opravdajte izbor numeričkim dokazima.

19.[20.] Vizualizirajte granice odlučivanja u 2D prostoru pomoću PCA za smanjenje dimenzionalnosti (samo za vizualizaciju) i opišite kako se granice mijenjaju s povećanjem složenosti modela.

Što predati – izvješće o vježbi

- Format: Google Colab s markdown ćelijama i ćelijama koda
- Mora sadržavati:
    a. Uvod (3–5 rečenica)
        I. Koji je bio problem? (izvorni i preformulirani)
        II. Zašto je to relevantno za znanosti o okolišu?
        III. Definirajte pozitivnu klasu i objasnite asimetrični trošak u ovom slučaju.
    b. Opis podataka
        I. Broj instanci, broj značajki (grupirano prema tipu)
        II. Ciljna varijabla i njezina distribucija
        III. Postupanje s nedostajućim podacima (napišite što ste učinili)
        IV. Odluka o poduzorkovanju
    c. Metode
        I. Koji su modeli trenirani i kako (LogReg + parametri)
        II. Podjela na treniranje/testiranje (navedite omjer)
        III. Bilo koje skaliranje i na kojem skupu podataka
    d. Rezultati
        I. Prikažite točnost, preciznost, odziv i F1, ROC-AUC i PR-AUC (u tabličnom formatu)
        II. Prikažite matricu zabune s objašnjenjem
        III. Navedite koji biste model odabrali i zašto (ne samo zato što „ima veću točnost”, nego možda zato što „ima veći odziv za pozitivnu klasu”)
    e. Rasprava
        I. Koji su rizici pogrešne klasifikacije u okolišnim primjenama u ovom primjeru?

II. Koja je uloga skaliranja?

III. Što biste sljedeće učinili da se ovo koristi za praćenje okoliša? (npr. podešavanje praga / kompromis preciznost–odziv, važnost značajki / interpretabilnost)

f. Zaključak

I. Nekoliko rečenica koje sažimaju što ste naučili iz ove vježbe s pomoću stvarnih okolišnih podataka – spomenite generalizaciju, izbor metrike i kompromise.

Vrednovanje

- **Kviz (10 pitanja): bit će osmišljen za procjenu konceptualnog razumijevanja i primijenjenog zaključivanja stečenog kroz predavanje i praktičnu vježbu o klasifikacijskim tehnikama u znanostima o okolišu. Procjenjivat će razumiju li studenti nadzirano učenje, granice odlučivanja (linearne nasuprot nelinearnima), osnovne evaluacijske metrike klasifikacije (točnost, preciznost, odziv, F1-mjera), ulogu podjele na treniranje/testiranje, fenomene podnaučenosti i prenaučenosti te važnost vizualizacije podataka.**

- **Praktični klasifikacijski zadatak: predviđanje razine prekoračenja NO_2 - Ova individualna praktična vježba studentima daje priliku za praktičan rad na klasifikacijskom problemu iz scenarija praćenja okoliša koji je obrađen u predavanju. Koristeći skup podataka o kvaliteti zraka, studenti treniraju i evaluiraju osnovni linearni model (logističku regresiju) kako bi utvrdili prelaze li razine dušikova dioksida (NO_2) definirani sigurnosni prag. Zadatak je osmišljen za procjenu sposobnosti studenata da provedu cjelovit tijek rada strojnog učenja, uključujući pripremu podataka, rukovanje značajkama, treniranje modela i evaluaciju. Poseban naglasak stavlja se na interpretaciju lažno pozitivnih i lažno negativnih ishoda u kontekstu potpore odlučivanju o kvaliteti zraka, pokazujući kako klasifikacijski modeli mogu pomoći praćenju okoliša i procjeni rizika.**

- **Praktični klasifikacijski zadatak: detekcija staništa (Spruce/Fir) - Ova individualna praktična vježba pruža studentima priliku da praktično primijene klasifikacijske modele na okolišni slučaj uporabe uveden u predavanju. Koristeći skup podataka o zemljišnom pokrovu s geografskim i okolišnim varijablama, studenti grade i evaluiraju osnovni linearni**

klasifikator (logističku regresiju) za detekciju prisutnosti specifičnog tipa staništa (šuma Spruce/Fir) nasuprot svim ostalim tipovima zemljišnog pokrova. Zadatak je osmišljen za procjenu sposobnosti studenata da provedu cjelovit tijek rada strojnog učenja, uključujući pripremu podataka, rukovanje značajkama, treniranje modela i evaluaciju. Poseban naglasak stavlja se na interpretaciju lažno pozitivnih i lažno negativnih ishoda u kontekstu potpore okolišnom odlučivanju, pokazujući kako klasifikacijski modeli mogu pomoći kartiranju staništa, planiranju zaštite i analizi korištenja zemljišta.

- **Kratko pisano izvješće: kratko pisano izvješće predstavlja završni korak praktične vježbe detekcije staništa. Njegova je svrha omogućiti studentima da sažmu, interpretiraju i jasno komuniciraju rezultate dobivene iz njihovih klasifikacijskih modela primijenjenih na problem detekcije staništa Spruce/Fir. Kroz ovo izvješće od studenata se očekuje da pokažu sposobnost:**
  - jasnog objašnjavanja koraka svojega klasifikacijskog tijeka rada, od pripreme podataka do evaluacije modela,
  - interpretacije klasifikacijskih metrika (točnost, preciznost, odziv, F1-mjera i matrica zabune) u kontekstu detekcije staništa i okolišnog odlučivanja, i
  - promišljanja o odabiru modela (linearni nasuprot nelinearnom), uz raspravu o tome kako složenost modela utječe na uspješnost i što to znači za stvarne primjene kao što su kartiranje staništa, planiranje zaštite i analiza korištenja zemljišta.

Kriteriji vrednovanja

ZA KVIZ

- Kviz će procjenjivati činjenično, konceptualno i temeljno teorijsko razumijevanje klasifikacije u ML-u, uključujući:
  - paradigmu nadziranog učenja,
  - razliku između linearnih i nelinearnih granica odlučivanja,
  - ulogu podjele na treniranje/validaciju/testiranje,
  - klasifikacijske metrike i kada ih koristiti
  - primjere okolišnih klasifikacijskih problema (prekoračenje NO2 i detekcija staništa)

Prag prolaznosti: 50 % točnih odgovora.

ZA PRAKTIČNI KLASIFIKACIJSKI ZADATAK

- Priprema i predobrada podataka
  - Ispravno učitavanje i pregled skupa podataka (oblik, ciljna varijabla, nedostajuće vrijednosti).
  - Odgovarajuće postupanje s nedostajućim podacima (izbacivanje ili imputacija) te, gdje je potrebno, skaliranje/normalizacija značajki.
  - Transparentan i reproducibilan tijek predobrade (kod je čist, koraci su jasni).

- Implementacija modela
  - Točno postavljanje i treniranje najmanje dvaju klasifikacijskih modela:
    - jednog osnovnog (npr. logistička regresija),
    - drugog nelinearnog
  - Odgovarajuća uporaba podjele na treniranje/testiranje (i validaciju ako se koristi).
  - Kod je organiziran, a parametri su izričito navedeni.

- Evaluacija i dijagnostika modela
  - Ispravan izračun i izvještavanje klasifikacijskih metrika:
    - točnost
    - preciznost
    - odziv
    - F1-mjera
    - matrica zabune
    - Logična usporedba modela (npr. „logistička regresija poboljšala je odziv za pozitivnu klasu u usporedbi s nelinearnim modelom dobivenim inženjeringom značajki”).
    - Svijest o neuravnoteženosti klasa i njezinim učincima na točnost.

- Odabir modela i obrazloženje
  - Jasno obrazloženje koji se model preferira (ne samo „veća točnost”, nego, gdje je relevantno, „bolji odziv za manjinsku / okolišno kritičnu klasu”).
  - Razumno objašnjenje sposobnosti generalizacije modela na temelju rezultata testnog skupa podataka

- Vizualizacija i prezentacija rezultata
  - Matrica zabune je nacrtana, označena i interpretirana.
  - Odabir najboljeg modela temelji se na stvarnim podacima iz tablice ili grafa
  - Vizualni prikazi su čitljivi i podupiru argument.

- Povezanost s okolišnim kontekstom
  - Student pokazuje razumijevanje pogrešne klasifikacije u kontekstu detekcije staništa, prepoznajući da je pogrešno klasificiranje područja koja nisu stanište kao staništa Spruce/Fir općenito ozbiljnije od propuštanja stvarne instance staništa.
  - Student može objasniti kako takav klasifikator može poduprijeti praćenje okoliša, planiranje zaštite i upravljanje korištenjem zemljišta tako što pomaže identificirati i pratiti rasprostranjenost staništa Spruce/Fir, prioritetizirati područja za zaštitu ili terensku provjeru te informirati okolišno odlučivanje vođeno podacima.

ZA KRATKO PISANO IZVJEŠĆE

- Jasnoća i struktura izvješća
  - Izvješće je logično organizirano (Uvod → Podaci → Metode → Rezultati → Rasprava → Zaključak).
  - Koraci tijeka rada vidljivi su u bilježnici
  - Jezik je precizan i sažet.

- Interpretacija rezultata i rasprava
  - Klasifikacijski rezultati (točnost, preciznost, odziv, F1, matrica zabune) točno su prikazani i smisleno interpretirani.
  - Student pokazuje razumijevanje pogrešaka po klasama iz matrice zabune.
  - Student promišlja zašto je jedan model bio bolji od drugoga
  - Rasprava pokazuje svijest o prenaučenosti i podnaučenosti, kao i o ulozi validacije.

- Povezanost s ekološkim i okolišnim kontekstom
  - Nalazi su povezani sa stvarnim okolišnim značenjem (npr. „veći odziv za Spruce/Fir važniji je od ukupne točnosti”).

- Student pokazuje razumijevanje kako klasifikacija vođena podacima može pridonijeti praćenju okoliša i informiranom odlučivanju.

- Kvaliteta vizualne prezentacije
  - Slike (matrica zabune, evaluacijske metrike, usporedba modela) jasno su označene.
  - Vizualni prikazi referencirani su u tekstu („kao što je prikazano na Slici 1…").
  - Vizualni prikazi pomažu objasniti ponašanje modela, a ne služe samo kao ukras izvješću.

- Komunikacija i stil
  - Pisanje je jasno, objektivno i profesionalno.
  - ML i okolišna terminologija koriste se ispravno (klasa, oznaka, značajka, preciznost, odziv, pozitivna klasa).
  - Student priznaje ograničenja (mali skup podataka, neuravnoteženost i nedostatak prostorne komponente), izazove koje uzrokuju visokodimenzionalni podaci te potrebu za poduzorkovanjem velikih skupova podataka pri radu u besplatnom okruženju Google Colab.

KVIZ (10 pitanja)

1. Nadzirano učenje

Koja od sljedećih tvrdnji najbolje opisuje nadzirano učenje u kontekstu analize okolišnih podataka?

A. Model otkriva obrasce u podacima bez unaprijed definiranih oznaka
B. Model uči preslikavanje iz ulaznih značajki u poznate ciljne oznake
C. Model automatski grupira slična okolišna opažanja
D. Model optimizira pravila odlučivanja bez korištenja podataka za treniranje

2. Binarna preformulacija

Zašto je izvorni višeklasni problem šumskog pokrova preformuliran u zadatak binarne klasifikacije?

A. Logistička regresija ne može se koristiti za višeklasne probleme
B. Kako bi se pojednostavnila interpretacija metrika i granica odlučivanja

C. Skup podataka sadrži samo dva smislena tipa pokrova

D. Kako bi se izbjeglo skaliranje značajki

3. Definicija pozitivne klase (detekcija staništa)

Što pozitivna klasa (1) predstavlja u zadatku detekcije staništa Spruce/Fir?

A. Bilo koje šumovito područje

B. Područja s velikom nadmorskom visinom

C. Lokacije pokrivene šumom Spruce/Fir

D. Lokacije s minimalnim ljudskim utjecajem

4. Lažno pozitivni ishodi u detekciji NO_2

Što predstavlja lažno pozitivan ishod u zadatku klasifikacije prekoračenja NO_2?

A. Ispravno identificiranje nesigurne kvalitete zraka

B. Predviđanje sigurne kvalitete zraka kada NO_2 prelazi prag

C. Predviđanje prekoračenja NO_2 kada je kvaliteta zraka zapravo sigurna

D. Neuspjeh u treniranju modela

5. Ozbiljnost pogrešne klasifikacije (detekcija staništa)

Koja se pogrešna klasifikacija u zadatku detekcije staništa obično smatra okolišno rizičnijom?

A. Lažno negativan ishod

B. Lažno pozitivan ishod

C. Istinito negativan ishod

D. Istinito pozitivan ishod

6. Odabir metrike

Koja je metrika najprikladnija ako je primarni cilj detektirati što više stvarnih staništa Spruce/Fir?

A. Točnost

B. Preciznost

C. Odziv

D. Specifičnost

7. Preciznost nasuprot odzivu

U kojoj bi situaciji preciznost bila važnija od odziva?

A. Kada je stanište vrlo skupo
B. Kada lažni alarmi dovode do skupih terenskih inspekcija
C. Kada je skup podataka savršeno uravnotežen
D. Kada se vizualiziraju granice odlučivanja

8. Podjela na treniranje/testiranje

Koja je glavna svrha podjele podataka na skup za treniranje i testni skup?

A. Smanjiti veličinu skupa podataka
B. Podesiti hiperparametre
C. Procijeniti generalizaciju na neviđene podatke
D. Poboljšati vizualizaciju

9. Podnaučenost nasuprot prenaučenosti

Model ima vrlo dobru uspješnost na podacima za treniranje, ali lošu uspješnost na testnim podacima. To je primjer:

A. Podnaučenosti
B. Prenaučenosti
C. Neuravnoteženosti klasa
D. Nasumične pogreške

10. Vizualizacija granice odlučivanja

Zašto su granice odlučivanja u praktičnim vježbama vizualizirane u 2D pomoću PCA?

A. Kako bi se poboljšala točnost modela
B. Kako bi se smanjila prenaučenost
C. Kako bi ponašanje visokodimenzionalnog modela bilo interpretabilno
D. Kako bi se zamijenile evaluacijske metrike

Predloženo ponderiranje

- Kviz: 20 %

- Praktični klasifikacijski zadatak (bilježnica/kod): 50 %
- Kratko pisano izvješće: 30 %

Reference / dodatna literatura

[1] C. O. De Burgh-Day and T. Leeuwenburg, “Machine learning for numerical weather and climate modelling: A review,” Nov. 14, 2023, *Copernicus Publications*. doi: 10.5194/gmd-16-6433-2023.

[2] M. Karmoude *et al.*, “Machine learning for air quality prediction and data analysis: Review on recent advancements, challenges, and outlooks,” Nov. 01, 2025, *Elsevier B.V.* doi: 10.1016/j.scitotenv.2025.180593.

[3] G. O. Ojwang *et al.*, “An integrated hierarchical classification and machine learning approach for mapping land use and land cover in complex social-ecological systems,” *Frontiers in Remote Sensing*, vol. 4, 2023, doi: 10.3389/frsen.2023.1188635.

[4] K. S. Bobba, K. K, V. K. Sai, D. Bugga, and V. M. S. Bolla, “AI-Enhanced Acoustic Analysis for Comprehensive Biodiversity Monitoring and Assessment,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.12897

[5] T. M. . Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[6] A. E. Maxwell, T. A. Warner, and F. Fang, “Implementation of machine-learning classification in remote sensing: An applied review,” May 03, 2018, *Taylor and Francis Ltd.* doi: 10.1080/01431161.2018.1433343.

[7] M. J. Kochenderfer *et al.*, *Decision Making Under Uncertainty*. The MIT Press, 2015. doi: 10.7551/mitpress/10187.001.0001.

[8] F. K. Ewald, L. Bothmann, M. N. Wright, B. Bischl, G. Casalicchio, and G. König, “A Guide to Feature Importance Methods for Scientific Inference,” Aug. 2024, doi: 10.1007/978-3-031-63797-1_22.

[9] Partha Protim Roy, Md. Shahriar Abdullah, and Iqtiar Md. Siddique, “Machine learning empowered geographic information systems: Advancing Spatial analysis and decision making,” *World Journal of Advanced Research and Reviews*, vol. 22, no. 1, pp. 1387–1397, Apr. 2024, doi: 10.30574/wjarr.2024.22.1.1200.

[10] S. Zhong *et al.*, “Machine Learning: New Ideas and Tools in Environmental Science and Engineering,” *Environ. Sci. Technol.*, p. acs.est.1c01339, Aug. 2021, doi: 10.1021/acs.est.1c01339.

[11] T. F. Deressu, A. K. Bojer, T. G. Debelee, W. G. Negera, S. Nadarajah, and K. W. Gebissa, “Enhancing land use and land cover classification with deep learning-based satellite imagery segmentation,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 144, Nov. 2025, doi: 10.1016/j.jag.2025.104839.

[12] A. Elsayed, J. Levison, A. Binns, M. Larocque, and P. Goel, “Regression-based machine learning models for nitrate and chloride prediction in surface water in a small agricultural sand plain sub-watershed in southwestern Ontario, Canada,” *Front. Environ. Sci.*, vol. 13, 2025, doi: 10.3389/fenvs.2025.1543852.

[13] T. Vo-Van, A. Nguyen-Hai, M. V. Tat-Hong, T. Nguyen-Trang, and F. Gomariz, “A New Clustering Algorithm and Its Application in Assessing the Quality of Underground Water,” *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/6458576.

[14] *Machine Learning in Earth, Environmental and Planetary Sciences*. Elsevier, 2023. doi: 10.1016/C2022-0-00549-3.

[15] F. Pedregosa FABIANPEDREGOSA *et al.*, “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot,” 2011. [Online]. Available: http://scikit-learn.sourceforge.net.

[16] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, Inc., 2023.