Tema: Uvod v strojno učenje

Predmet: Strojno učenje za okoljske aplikacije

Opis učne enote:

Ta uvodna učna enota podaja osnovni pregled strojnega učenja (ML) in njegove vloge pri reševanju okoljskih in ekoloških problemov. Predstavljene so glavne naloge strojnega učenja (razvrščanje, regresija, gručenje) ter strategije učenja (nadzorovano in nenadzorovano učenje). Poudarek je na razumevanju ključnih konceptov, vključno s pripravo podatkov, oblikovanjem značilk, učenjem in vrednotenjem modelov, ponazorjenih z jasnimi vizualnimi primeri. Podrobno sta pojasnjena problema podprileganja in prenaučenja ter njune posledice. Med okoljskimi primeri uporabe sta razvrščanje ravni kakovosti zraka in zaznavanje habitatov.

Učni izidi:

Po zaključku te enote bodo študenti znali:

1. Pojasniti, kaj je strojno učenje, in opisati njegove glavne uporabe v okoljskih znanostih.
2. Razlikovati med nadzorovanim in nenadzorovanim učenjem.
3. Prepoznati osnovne naloge strojnega učenja, kot so regresija, razvrščanje in gručenje.
4. Razumeti osnovni potek projekta strojnega učenja (priprava podatkov, učenje, vrednotenje).
5. Prepoznati in pojasniti pogoste težave pri učenju modelov strojnega učenja, kot so prenaučenje, podprileganje in težave s kakovostjo podatkov.
6. Naučiti in ovrednotiti preprost model strojnega učenja z uporabo okoljskih podatkov.

Uvod

Strojno učenje (ML) računalnikom omogoča učenje iz podatkov brez potrebe po izrecnem programiranju. V zadnjih letih se strojno učenje široko uporablja pri reševanju različnih okoljskih problemov, pri čemer dopolnjuje ali v nekaterih primerih nadomešča tradicionalne pristope okoljskega modeliranja, ki temeljijo predvsem na formalnih fizikalnih in kemijskih enačbah ter njihovem povezovanju z drugimi formalizmi v biologiji, geologiji in ekologiji. Z naraščanjem obsega in raznolikosti okoljskih podatkov (senzorji, naprave interneta stvari, obsežni sistemi spremljanja) se je poudarek premaknil k podatkovno vodenim metodam, ki se učijo neposredno iz zbranih podatkov.

Strojno učenje je podatkovno vodena disciplina in eno najpomembnejših področij umetne inteligence. Namesto vnaprejšnje določitve nespremenljive strukture modela algoritmi strojnega učenja iz primerov sklepajo o osnovnih povezavah, s čimer znanstvenikom in strokovnjakom omogočajo modeliranje sistemov, pri katerih so eksplicitni fizikalni zakoni neznani, nepopolni ali preveč zapleteni.

Okoljski pojavi (npr. podnebni procesi, degradacija ekosistemov in dinamika onesnaževanja) so po svoji naravi kompleksni, nelinearni in večrazsežni problemi. Področje strojnega učenja ponuja različna orodja in pristope za obravnavo tako zapletenih vprašanj. Ker so za te probleme na voljo obsežni podatki, jih je poleg tega s strojnim učenjem pogosto mogoče obravnavati učinkoviteje kot s tradicionalnimi regresijskimi ali simulacijskimi modeli.

Nekaj primerov uporabe strojnega učenja v okoljskih znanostih:

- Napovedovanje podnebja in vremena: modeli strojnega učenja prepoznavajo vzorce v dolgoletnih meteoroloških zapisih in satelitskih opazovanjih ter napovedujejo temperaturne anomalije ali skrajne padavinske dogodke [1].
- Modeliranje kakovosti zraka in vode: napovedni modeli se učijo povezav med emisijami, meteorološkimi spremenljivkami in koncentracijami onesnaževal [2].
- Daljinsko zaznavanje in razvrščanje rabe tal: slikovni modeli strojnega učenja samodejno razvrščajo vrste pokrovnosti tal iz multispektralnih satelitskih podatkov [3].
- Spremljanje biotske raznovrstnosti in ekosistemov: strojno učenje pomaga pri modeliranju razširjenosti vrst, akustičnem razvrščanju in zaznavanju ekoloških anomalij [4].

Omenili smo že, da je strojno učenje sistem, ki se uči iz podatkov, ne da bi ga bilo treba izrecno programirati za obravnavano nalogo.

> *»Za računalniški program pravimo, da se uči iz izkušenj E glede na neki razred nalog T in mero uspešnosti P, če se njegova uspešnost pri nalogah iz T, merjena s P, izboljšuje z izkušnjami E.« [5]*

Ta opredelitev poudarja tri ključne sestavine:

- Podatki (izkušnje E): temelj, iz katerega se model uči vzorcev.
- Naloga (T): kar naj bi model opravil, na primer razvrščanje, napovedovanje ali gručenje.
- Uspešnost (P): kvantitativna mera kakovosti delovanja modela (točnost, RMSE, ocena F1 itd.).

Primer: razviti želimo aplikacijo, ki lahko na slikah zazna avtomobile. Če bi želeli vnaprej zapisati nabor pravil (ČE-POTEM), ki opisujejo avtomobil, na primer:

ČE je na sliki predmet s 4 kolesi, pravokotne oblike in z vetrobranskim steklom, POTEM je to AVTOMOBIL; sicer to NI AVTOMOBIL.

Ta pristop hitro odpove, saj so avtomobili iz različnih zornih kotov videti različno, osvetlitev vpliva na prepoznavanje, nekateri predmeti pa so avtomobilom podobni. Zato pravimo, da je ta pristop preveč tog in ne daje dobrih rezultatov.

Iz več virov smo izvedeli, da lahko pristop strojnega učenja reši tovrstni problem, če imamo na voljo razmeroma veliko količino podatkov.
Zato smo se odločili uporabiti to strategijo: zbrali smo precejšnje število slik avtomobilov v različnih okoliščinah in enako število slik brez avtomobilov. Prebrali smo, da je za to nalogo priporočljiv določen model strojnega učenja za prepoznavanje slik. Vse slike so opremljene z ustreznimi in pravilnimi oznakami (CAR/NO CAR). Naloga modela strojnega učenja je napovedati pravilno oznako slike. Model se te naloge uči iz slik (podatkov), lahko pa rečemo tudi, da se uči iz izkušenj (s prikazovanjem slik). Sprva bo model v številnih primerih neuspešen, vendar se bo sčasoma (z več podatki oziroma izkušnjami) naučil pravilno napovedati oznake v večini primerov. Odstotek pravilno napovedanih oznak slik je mera uspešnosti, ki kaže, kako dobro se je model naučil naloge.

Za model pravimo, da se **uči**, ker se njegova **uspešnost** pri zaznavanju avtomobilov na slikah **izboljšuje**, ko **pridobiva več izkušenj s podatki**!

Običajen potek strojnega učenja vključuje:

1. Zbiranje in predobdelavo podatkov
2. Oblikovanje in predstavitev značilk
3. Izbiro in učenje modela

4. Vrednotenje in preverjanje veljavnosti
5. Interpretacijo in uvedbo v uporabo

Poleg zanesljive napovedne uspešnosti v okoljskih znanostih [6] strojno učenje zagotavlja temelj za kakovostne sisteme odločanja v pogojih negotovosti [7] ter ponuja razložljiv vpogled v pomembnost značilk [8], prostorske odvisnosti in občutljivost sistema [9]. Ker količina okoljskih podatkov še naprej eksponentno narašča, postaja poznavanje strojnega učenja nepogrešljiv del sodobnega upravljanja okoljskih podatkov [10].

Pogled v ozadje osnov strojnega učenja

Za okoljske znanosti je značilna velika količina podatkov iz raznovrstnih virov, zato je smiselno na tem področju izkoristiti uporabnost modelov strojnega učenja. Strojno učenje dopolnjuje običajne znanstvene pristope z odkrivanjem skritih vzorcev in sklepanjem o povezavah. V tem uvodnem predavanju se ne osredotočamo na podrobne razlage modelov za različne naloge strojnega učenja, saj bodo obravnavani v naslednjih predavanjih, temveč na temeljna načela, ki veljajo za vse naloge. Glavni cilj je konceptualno razumevanje, ne matematični formalizem delovanja modelov strojnega učenja, ki bo obravnavan v predavanjih, namenjenih posameznim nalogam strojnega učenja.

Nadzorovano in nenadzorovano učenje

Iz opredelitve strojnega učenja je razvidno, da se lahko modeli strojnega učenja naučijo opravljati nalogo z učenjem iz podatkov. Obstajata dve osnovni vrsti učenja: nadzorovano in nenadzorovano. Nadzorovano učenje je strategija, pri kateri model učimo na podatkih z znanimi vhodnimi vrednostmi in pripadajočimi izhodi. Primer je razvrščanje satelitskih slik v osem kategorij pokrovnosti tal: pozidana območja, ceste, vodna telesa, kmetijska zemljišča, grmišča, gozdovi, travišča in drugo [11]. Oznake kategorij pridobimo od področnih strokovnjakov; za vsako sliko v podatkovni množici vnaprej poznamo oznako, model pa je ne. Z uporabo ustrezne mere uspešnosti (npr. absolutne točnosti) se lahko model nauči napovedovati te kategorije na označenih slikah in, kar je še pomembneje, na novih, prej nevidenih slikah. Izraz nadzorovano se tukaj nanaša na pripise (oznake), pridobljene od nadzornikov oziroma področnih strokovnjakov, ki pri učenju naloge določajo referenčno resnico. Poleg omenjenega razvrščanja je druga značilna naloga nadzorovanega učenja regresija, pri kateri se model strojnega učenja uči iz vhodnih vrednosti z znanimi realnoštevilskimi izhodi. Primer je napovedovanje koncentracij nitratnega dušika in klorida iz nabora neodvisnih spremenljivk (povprečni dnevni površinski pretok, padavine v 24h pred vzorčenjem, lokacija vzorčnega mesta, pokrovnost s poljščinami, temperatura, pH, raztopljeni kisik, oksidacijsko-redukcijski potencial in električna prevodnost). Odvisne spremenljivke so kazalniki kakovosti površinskih voda v kmetijskih porečjih [12]. Pri regresijskih nalogah je nadzor posreden in se izraža prek zbranih neodvisnih in odvisnih podatkov.

Pri nenadzorovanem učenju pa za opazovane vhode ni vnaprej natančno opredeljenega in znanega izhoda. Pravzaprav so znani samo vhodi. Model strojnega učenja mora v vhodnih podatkih poiskati pravilnosti in izpostaviti pomenljive vzorce s konkretnim pomenom na obravnavanem področju. Tu ni vpliva nadzornika, ki bi lahko določil referenčno resnico, zato to vrsto učenja na področju strojnega učenja imenujemo nenadzorovano učenje.

Značilna naloga v paradigmi nenadzorovanega učenja je gručenje, katerega cilj je podobne podatkovne primerke združiti v isto skupino, med seboj različne primerke pa razvrstiti v različne skupine (to je tako imenovani pristop trdega gručenja). Kadar lahko isti primerek pripada več kot eni skupini (gruči), z določeno stopnjo pripadnosti vsaki skupini, govorimo o pristopih mehkega gručenja. V okoljskih znanostih ima gručenje pomembno vlogo in ga je mogoče uporabiti za ocenjevanje kakovosti podzemne vode [13].

Za omenjene naloge je mogoče uporabiti številne modele strojnega učenja:

- Razvrščanje (SVM – stroji podpornih vektorjev, odločitvena drevesa, RIPPER, K-NN, umetne nevronske mreže, logistična regresija, naključni gozd, naivni Bayes, AdaBoost, Bagging, Stacking itd.)
- Regresija (linearna regresija, SVM za regresijo, odločitvena drevesa za regresijo, večslojni perceptron itd.)
- Gručenje (gručenje K-means, algoritem pričakovanja in maksimizacije, DBSCAN, HDBSCAN, Cobweb, hierarhično gručenje, mehki C-means itd.)

Večino teh modelov strojnega učenja boste podrobneje spoznali v naslednjih predavanjih, zato bomo tukaj podali le osnovni vpogled v model, ki ga potrebujemo za nadaljnjo predstavitev temeljnih konceptov strojnega učenja, kot so priprava podatkov, oblikovanje značilk, učenje, testiranje, vrednotenje uspešnosti ter pojava prenaučenja in podprileganja. Poleg tega bo na kratko predstavljen Google Colab.

Logistična regresija – temeljni vpogled

Čeprav vsebuje besedo *regresija*, **je logistična regresija model za razvrščanje**, ki se najpogosteje uporablja pri **problemih binarnega razvrščanja**. Ta vrsta razvrščanja, kot pove ime, obravnava problem z dvema razredoma, ki ju običajno označimo z 0 in 1.

Njen cilj ni napovedati zvezne vrednosti, temveč oceniti **verjetnost, da primer pripada danemu razredu** (običajno razredu 1).

Formalno modelira:

$P(y=1;x,\theta)$ Logično je, da je ob poznavanju te verjetnosti znana tudi druga:

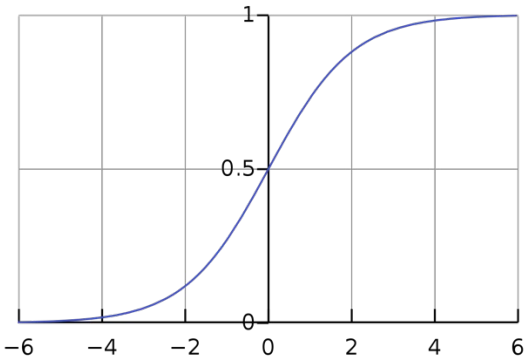
$$P(y=0;x,\theta)=1-\ P(y=1;x,\theta)$$

Pri čemer:

- **x** je vektor značilk
- y (0 ali 1) je oznaka razreda
- θ so parametri modela (uteži)

V jedru logistične regresije je **sigmoidna (logistična) funkcija**, ki preslika poljuben realnoštevilski vhod v interval **(0, 1)**:

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



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

To je ključno, ker:

- morajo biti verjetnosti med 0 in 1,
- Sigmoidna funkcija gladko pretvori linearno kombinacijo značilk ($\theta^T \cdot x$) v verjetnost.

Interpretacija je zdaj tesno povezana z verjetnostmi in ne le z oznakami. Logistična regresija vrne številski izhod:

- $P(y = 1 \vee x)$ — verjetnost pozitivnega razreda
- $P(y = 0 \vee x) = 1 - P(y = 1 \vee x)$ — verjetnost negativnega razreda

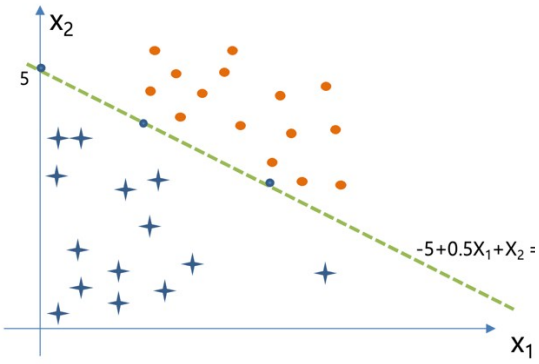
Zato je logistična regresija posebej uporabna, kadar:

- so razredi neuravnoteženi,
- je treba prilagoditi pragove odločanja,
- je pomembno razvrščanje po vrstnem redu ali ocenjevanje zanesljivosti.

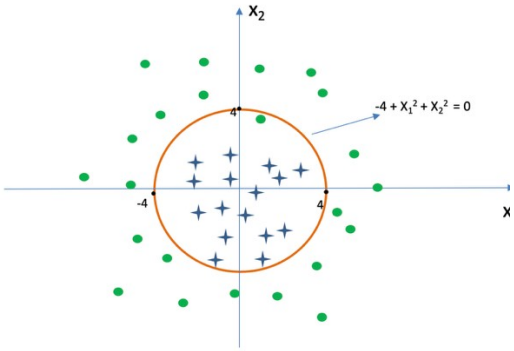
Pojavita se dve pomembni vprašanji:

- Po katerem načelu določimo uteži ($\theta_{i)?}$
- Kako je videti odločitvena meja tega modela?

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



Slika 2 Linearna odločitvena meja



Slika 3 Nelinearna odločitvena meja

Odločitev:→ vse točke nad črto se razvrstijo v razred 1, sicer pa v razred 0

Odločitev:→ vse točke zunaj kroga se razvrstijo v razred 1, sicer pa v razred 0

Slika 2 prikazuje prvi primer logistične regresije, ki v izvirnem prostoru značilk ustvari linearno odločitveno mejo. Slika 3 prikazuje nelinearno odločitveno mejo, ki jo dobimo šele po oblikovanju značilk, pri katerem izvirne vhodne značilke preoblikujemo (npr. z dodajanjem kvadratnih členov), medtem ko model ostaja linearen glede na parametre. Uteži θ se naučijo iz vhodnih podatkov z optimizacijo izbrane mere uspešnosti.

V tem predavanju mere uspešnosti in njihova podrobna razlaga niso v ospredju, saj bodo obravnavane v naslednjih predavanjih.

Točnost

Točnost meri delež pravilnih napovedi med vsemi napovedmi.

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

Točnost odgovarja na vprašanje: »Kako pogosto je model na splošno pravilen?«

Preciznost

Preciznost meri, koliko primerkov, napovedanih kot pozitivnih, je dejansko pozitivnih.

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

Preciznost odgovarja na vprašanje: »Kako pogosto ima model prav, ko napove pozitiven primer?«

Priklic (občutljivost)

Priklic meri, koliko dejansko pozitivnih primerkov model pravilno zazna.

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

Visok priklic je ključnega pomena, kadar spregled okoljskega dogodka (kot je prekoračitev onesnaženosti ali prisotnost habitata) pomeni veliko tveganje.

Ocena F1

Ocena F1 združuje preciznost in priklic v eno mero z uporabo njune harmonične sredine.

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

Ocena F1 je uporabna, kadar so pomembni tako lažno pozitivni kot lažno negativni rezultati in kadar je podatkovna množica neuravnotežena, kar je pogosto pri okoljskih aplikacijah.

Krivulja ROC

Krivulja karakteristike delovanja sprejemnika (ROC) prikazuje delež resnično pozitivnih rezultatov (priklic) v odvisnosti od deleža lažno pozitivnih rezultatov za vse mogoče pragove razvrščanja.

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

Krivulja ROC ponazarja kompromis med zaznavanjem resničnih okoljskih dogodkov in pojavnostjo lažnih alarmov.

ROC-AUC

ROC-AUC (ploščina pod krivuljo ROC) povzema krivuljo ROC z eno vrednostjo med 0 in 1.

Razlaga:

- 0.5 pomeni naključno ugibanje
- 1.0 pomeni popolno razločevanje

ROC-AUC izraža, kako dobro lahko model pozitivne primere uvrsti višje od negativnih neodvisno od izbranega praga odločanja.

Krivulja preciznost–priklic (PR)

Krivulja preciznost–priklic (PR) prikazuje preciznost glede na priklic pri različnih pragovih razvrščanja.

Krivulja PR neposredno pokaže, kako se preciznost spreminja z naraščanjem priklica, in izpostavi kompromis med lažnimi alarmi in spregledanimi zaznavami.

PR-AUC (povprečna preciznost)

PR-AUC povzema krivuljo preciznost–priklic z eno vrednostjo. PR-AUC je posebej informativen za okoljske podatkovne množice, v katerih so pozitivni primeri redki, kot so prekoračitve onesnaženosti ali določene vrste habitatov. Na lažno pozitivne rezultate je občutljivejši kot ROC-AUC.

Ključna ugotovitev

Nobena mera ni najboljša v vseh primerih. Pri strojnem učenju za okoljske namene mora izbira mere odražati prednostne naloge uporabe:

- Uporabite priklic, kadar je spregled resničnih dogodkov tvegan
- Uporabite preciznost, kadar so lažni alarmi dragi
- Uporabite oceno F1 za uravnoteženje obeh
- Pri zaznavanju redkih dogodkov dajte prednost PR-AUC in krivuljam PR
- Za splošno primerjavo modelov uporabite ROC-AUC

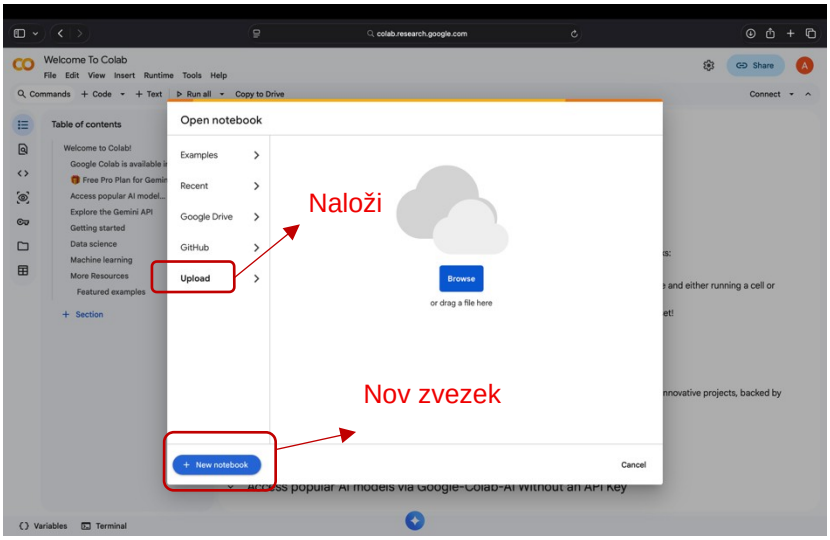
Google Colab

Google Colab je platforma v oblaku, ki omogoča brezplačen dostop prek brskalnika do Googlove računske zmogljivosti, vključno s CPU-ji, GPU-ji in TPU-ji. Temelji na ogrodju Jupyter Notebook ter uporabnikom omogoča pisanje in izvajanje kode Python brez kakršne koli lokalne namestitve. Vključuje vnaprej nameščene ključne knjižnice za strojno učenje in se brezhibno povezuje z Google Drive, vendar zahteva Googlov račun in deluje v celoti znotraj Googlovega ekosistema.

Storitev Google Colab je eden najpreprostejših načinov, da študenti izvajajo skripte Python, ne da bi morali ustvariti delovno okolje z vsemi potrebnimi knjižnicami. Preprosteje povedano, na uporabniku prijazen način zagotavlja okolje za izvajanje tudi zapletene kode Python, ki izpolnjuje vse zahteve in odvisnosti. Okolje vključuje osnovni tolmač Python ter dodatne knjižnice in module (vnaprej pripravljeno kodo, ki so jo razvili in preizkusili ugledni viri, da bi poenostavili in pospešili razvoj ter izboljšali zanesljivost). Te knjižnice podpirajo naloge, kot so učinkovito numerično računanje (NumPy), robustne

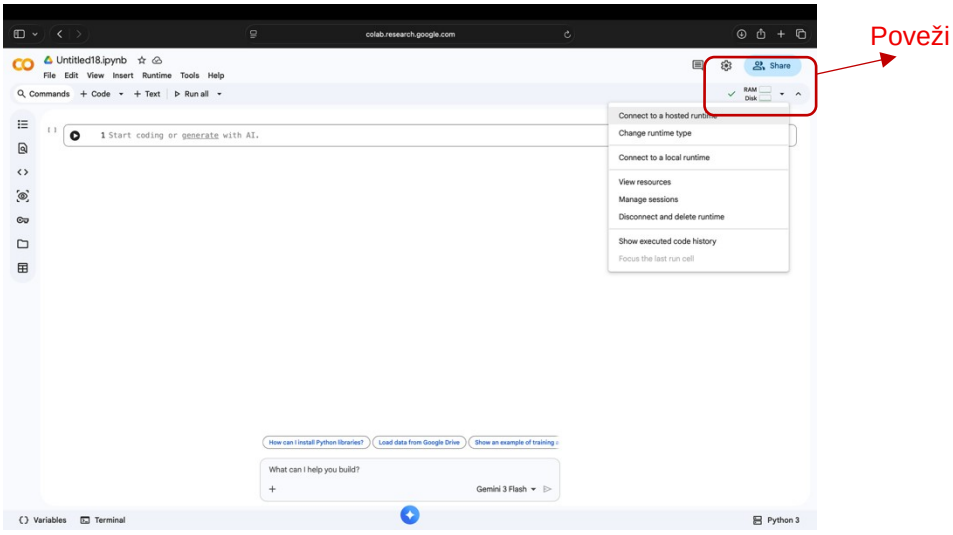
podatkovne strukture in analiza (Pandas), vizualizacija (Matplotlib) ter najsodobnejše izvedbe strojnega učenja (Scikit-learn, TensorFlow, PyTorch).

Za uporabo okolja Colab potrebujete Googlov račun. Če ga nimate, ga lahko ustvarite tukaj. Ko imate račun, lahko do okolja Colab dostopate tukaj. Prikazati bi se moral takšen zaslon:



Slika 4 Začetno okno okolja Google Colab

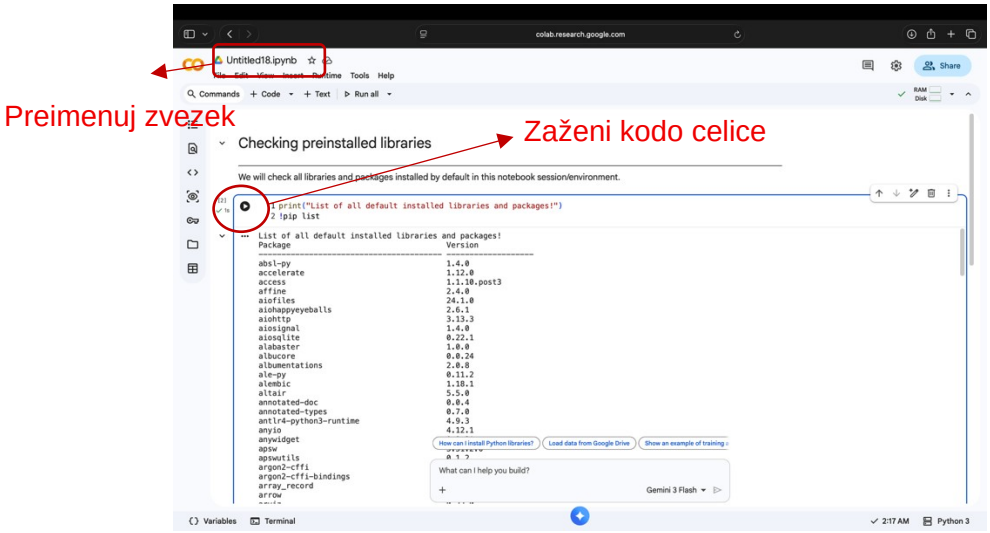
V tem oknu lahko odprete nov zvezek, naložite že pripravljenega ali odprete obstoječi zvezek. Enako lahko storite v meniju File. Za zdaj odprite nov zvezek.



Slika 5 Zagon novega zvezka Google Colab

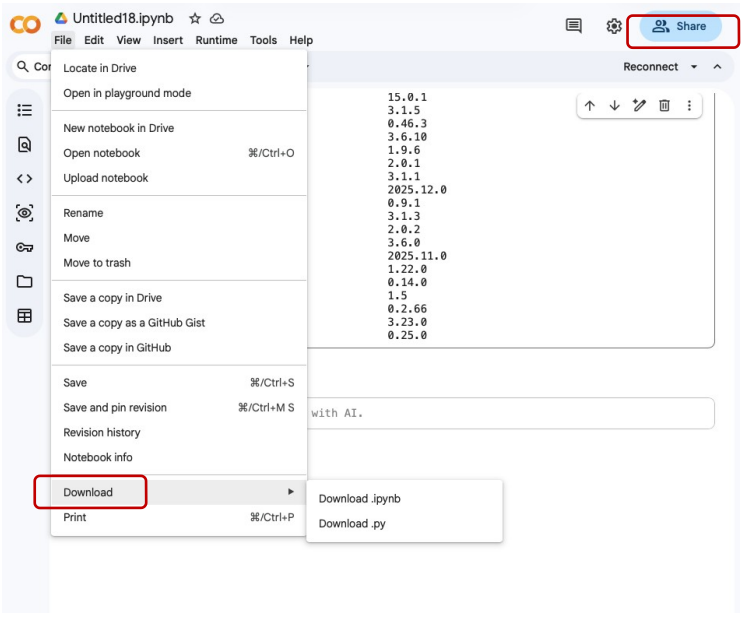
Slika 5 prikazuje postopek; v zgornjem desnem kotu kliknite puščico in izberite Connect to a hosted runtime. Ko je povezava vzpostavljena, bo kazalnik prikazal RAM, CPU in zeleno kljukico. Na voljo sta dve vrsti celic: kodne in besedilne celice (celice Markdown). Za izvajanje kode Python uporabite kodne celice. Celico lahko izvedete s klikom gumba

»play« levo od nje ali s pritiskom tipk »Shift+Enter« na tipkovnici; s tem se izbrana celica samodejno izvede in vstavi nova. Delo z besedilnimi celicami je preprosto: vstavite besedilno celico, jo dvokliknite, da preklopite v način urejanja, in začnite pisati. Na desni bo viden predogled, po kliku zunaj celice pa se prikaže rezultat. Za osnove oblikovanja besedila v teh celicah si oglejte vadnico za Markdown. Slika 6 prikazuje preimenovanje zvezka: v zgornjem levem kotu spremenite njegovo ime (tam, kjer je pripona ipynb).



Slika 6 Izvajanje kode Python v okolju Colab/preimenovanje zvezka

Zvezek Colab lahko shranite v Google Drive ali repozitorij GitHub, lahko pa ga tudi delite (v zgornjem desnem kotu izberite Share) ali prenesete kot datoteko .py oziroma .ipynb (v meniju File izberite možnost Download). Slika 7 *prikazuje postopek.*



Slika 7 Prenos zvezka/deljenje zvezka

Napovedovanje prekoračitve NO2 z uporabo podatkov senzorjev kakovosti zraka

Spremljanje kakovosti zraka ima ključno vlogo pri varovanju javnega zdravja in podpiranju okoljske politike. Sodobne merilne postaje neprekinjeno zbirajo velike količine podatkov kemijskih senzorjev in meteoroloških instrumentov. Zato je kakovost zraka odlično področje za prikaz, kako lahko strojno učenje podpre sprejemanje okoljskih odločitev v praksi.

V praksi okoljske odločitve le redko temeljijo na napovedovanju natančne vrednosti onesnaženosti. Namesto tega so ključna vprašanja navadno zelo preprosta:

- Ali je zakonsko določena mejna vrednost kakovosti zraka presežena?
- Ali je treba izdati opozorilo?
- Ali je treba aktivirati omilitvene ukrepe?

Ta primer uporabe se osredotoča na odgovarjanje na tovrstna vprašanja z napovedovanjem **dogodkov prekoračitve NO_2** s pomočjo strojnega učenja.

PODATKOVNA MNOŽICA za študijo primera

V tem predavanju uporabljamo **podatkovno množico Air Quality** iz repozitorija UCI Machine Learning Repository, znanega javnega repozitorija podatkovnih množic za strojno učenje.

- **Časovno obdobje:** od marca 2004 do februarja 2005

- **Pogostost vzorčenja:** ena meritev na uro
- **Lokacija:** urbana merilna postaja v italijanskem mestu
- **Število opazovanj:** približno 9300 urnih zapisov

Vsak zapis predstavlja okoljske razmere, izmerjene v eni uri.

Podatkovna množica združuje tri vrste meritev:

Referenčne koncentracije onesnaževal

Te vrednosti izvirajo iz referenčnih analizatorjev in predstavljajo zanesljive kemijske meritve:

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

V opisu izvirne podatkovne množice so te spremenljivke predvidene kot ciljne spremenljivke regresijskih modelov.

Signali senzorjev

Podatkovna množica vsebuje tudi odčitke senzorjev s kovinskimi oksidi. Ti senzorji so cenejši in bolj podvrženi šumu kot referenčni analizatorji, njihovi signali pa so pogosto nelinearni in navzkrižno občutljivi. V tem primeru uporabe so ti odčitki senzorjev **vhodne značilke** naših modelov.

Meteorološke spremenljivke

Vključene so okoljske razmere, ki vplivajo tako na ravni onesnaženosti kot na delovanje senzorjev:

- Temperatura
- Relativna vlažnost
- Absolutna vlažnost

Opombe o naravi podatkov časovnih vrst

Vse meritve imajo časovne žige in so zbrane zaporedno. To pomeni, da so sosednja opazovanja časovno povezana in niso povsem neodvisna.

V tem predavanju pa vsako meritev obravnavamo kot **posnetek stanja okolja v dani uri**. Da bi se izognili metodološkim težavam, podatke za učenje in testiranje razdelimo **kronološko**, s čimer posnemamo uporabo takih sistemov v praksi.

Opombe k izvirni opredelitvi problema

Po navedbah repozitorija UCI je ta podatkovna množica namenjena **regresiji**. Cilj izvirnega problema je bil iz signalov senzorjev in meteoroloških podatkov napovedati zvezne koncentracije onesnaževal (na primer NO_2).

Takšna formulacija je smiselna, kadar je cilj natančna številčna napoved.

Preoblikovanje problema za razvrščanje

Pri spremljanju okolja so natančne vrednosti onesnaževal pogosto manj pomembne od tega, ali je presežen kritični prag. Regulativni sistemi in opozorila za javnost se navadno sprožijo, ko onesnaženost preseže vnaprej določene mejne vrednosti.

Zato izvirni regresijski problem preoblikujemo v **nalogo binarnega razvrščanja**. Namesto napovedovanja natančne koncentracije NO_2 napovedujemo, ali pride do **dogodka prekoračitve**.

Uporabljeni regulativni prag

Prag, uporabljen v tej študiji primera, izhaja iz uradne evropske zakonodaje o kakovosti zraka:

Directive 2008/50/EC of the European Parliament and of the Council of 21 May 2008 on ambient air quality and cleaner air for Europe. Iz te direktive izhaja:

- **Urna mejna vrednost za NO_2 je 200 µg/m³**
- Ta mejna vrednost ne sme biti presežena več kot 18-krat na leto.

Uradno zakonodajno besedilo (EUR-Lex) za navedeno direktivo določa mejne vrednosti v ANNEX XI (Limit Values for the protection of Human health), v razdelku B.

Uporaba tega praga zagotavlja, da je naša naloga razvrščanja **zakonsko utemeljena in okoljsko smiselna**.

Opredelitev dveh razredov

Na podlagi predpisa EU ciljno spremenljivko opredelimo tako:

- **Razred 0 – sprejemljiva kakovost zraka:**
  koncentracija NO_2 pod 200 µg/m³
- **Razred 1 – dogodek prekoračitve:**
  koncentracija NO_2, ki je enaka ali višja od 200 µg/m³

Tako dobimo dva jasno določena in razložljiva razreda.

Zakaj je logistična regresija primerna za to uvodno predavanje

Pri tej preoblikovani formulaciji z logistično regresijo na podlagi razpoložljivih senzorskih in meteoroloških podatkov ocenjujemo verjetnost, da pride do prekoračitve NO_2.

Logistična regresija je za to nalogo zelo primerna, ker:

- zagotavlja **ocene verjetnosti** in ne le oznak razredov,
- je njene odločitvene meje preprosto razložiti,
- omogoča jasen prikaz **oblikovanja značilk**,
- omogoča preprosto opazovanje prenaučenja in podprileganja,
- podpira nazoren prikaz vedenja modela.

Narava podatkovne množice časovnih vrst v okviru preoblikovanja problema

Čeprav se podatki zbirajo skozi čas, to v predstavljenem primeru uporabe ne povzroča težav.

V tem predavanju:

- časovne dinamike ne modeliramo izrecno,
- vsako opazovanje obravnavamo kot neodvisen posnetek stanja,
- uporabimo kronološko razdelitev na učno in testno množico, da preprečimo uhajanje informacij.

Tehnike modeliranja časovnih vrst so iz te enote namenoma izključene, da se ohrani osredotočenost na temeljne koncepte strojnega učenja.

Tako nam prilagojeni primer uporabe omogoča raziskovanje širokega nabora pomembnih konceptov strojnega učenja v realističnem okoljskem okviru:

- preoblikovanje regresijske podatkovne množice v nalogo razvrščanja z uporabo veljavnih direktiv zakonodajnih organov,
- oblikovanje značilk na podlagi okoljskega znanja,
- razumevanje linearnih in nelinearnih odločitvenih mej,
- pravilno ločevanje podatkov za učenje in testiranje,
- prepoznavanje prenaučenja in podprileganja,
- uporaba prikazov za razlago in analizo modelov.

Zato ta primer uporabe vzpostavlja jasno in praktično povezavo med teorijo strojnega učenja ter dejanskimi okoljskimi uporabami **[14]**.

Pri mejnih vrednostih iz direktive se osredotočimo na razvrščanje prekoračitev po posameznih urah. Najprej bomo naložili podatke s paketom ucimlrepo. Druge knjižnice in paketi, potrebni v tej predstavitvi, so scikit-learn [15], pandas, numpy in 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 knjižnice scikit-learn, metrike pa so standardne mere za vrednotenje uspešnosti modela. Nastavitev naključnega stanja zagotavlja ponovljivost rezultatov. DummycCassifier je preprost klasifikator, ki vse primere dodeli večinskemu razredu (najpogostejšemu razredu) – potrebujemo ga za prikaz težav z nekaterimi standardnimi metrikami pri tako imenovanih problemih neuravnoteženega razvrščanja (ta primer uporabe sodi mednje). Podatke naložimo z ucimlrepo brez neposrednega prenosa datotek.

[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

```

Čiščenje manjkajočih vrednosti in razčlenjevanje časovnih žigov sta del postopka priprave podatkov. V ta namen bomo:

- združili **Date** in **Time** v časovni žig
- zamenjali **-200** z NaN
- odstranili vrstice, v katerih manjka referenčna vrednost NO_2 (potrebujemo jo za oznake)
- (neobvezno) odstranili stolpce s preveč manjkajočimi vrednostmi

[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 po čiščenju:

	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

Za preoblikovanje izvirnega regresijskega problema v problem razvrščanja bomo uporabili ravni prekoračitve iz navedene direktive EU. Ustvarimo oznake:

- **y = 1** če je $NO_2 \geq 200$ µg/m³
- **y = 0** sicer

To ustreza dejanskemu regulativnemu pragu, zato imajo razredi praktičen pomen.

[6] ✓ 0 s

```
df["y_exceed"] = (df[NO2_COL] >= EU_NO2_THRESHOLD).astype(int)

pos_rate = df["y_exceed"].mean()
print(f"Positive class rate (NO₂ exceedance): {pos_rate:.4f} ({pos_rate*100:.2f}%)")
df[["timestamp", NO2_COL, "y_exceed"]].head()

```

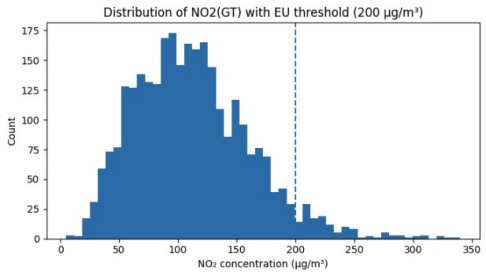
```
Positive class rate (NO₂ exceedance): 0.0478 (4.78%)
```

	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 zgornjih podatkov je razvidno, da razredu 1 (raven prekoračitve) pripada le 4.78% vseh podatkovnih točk, kar je značilen primer neuravnoteženega razvrščanja. Neuravnoteženost je še jasneje vidna iz prikaza porazdelitve NO2 glede na prag EU.

[22] ✓ 0 s

```
# NO2 distribution with EU threshold line
EU_NO2_THRESHOLD = 200.0  # µg/m³

plt.figure(figsize=(7,4))
plt.hist(df[NO2_COL].dropna(), bins=50)
plt.axvline(EU_NO2_THRESHOLD, linestyle="--")
plt.title(f"Distribution of {NO2_COL} with EU threshold (200 µg/m³)")
plt.xlabel("NO₂ concentration (µg/m³)")
plt.ylabel("Count")
plt.tight_layout()
plt.show()

# Class balance preview (will define label in next section)


```



Pripravo podatkov nadaljujemo z nadomeščanjem manjkajočih vrednosti z mediano, oblikovanje značilk pa z njihovo izbiro: odstranimo stolpce Date, Time in časovne žige, ki niso značilke, obdržimo le številske senzorske in meteorološke spremenljivke ter nazadnje odstranimo redko zapolnjene stolpce.

[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

Zdaj lahko pokažemo, zakaj je lahko sama točnost pri izrazito neuravnoteženih podatkovnih množicah zavajajoča. Pri pragu EU (NO_2 ≥ 200 µg/m³) je pozitivni razred navadno redek (gre za problem zaznavanja dogodkov). Zato je lahko zgolj točnost zavajajoča: model, ki vedno napove »brez prekoračitve«, lahko doseže visoko točnost, čeprav je neuporaben. Izračunamo izhodiščno napoved z večinskim razredom (z uporabo DummyClassifier) ter uvedemo preciznost/priklic in PR-AUC (povprečna preciznost) kot primernejše metrike za redke dogodke. Rezultat tega nazornega primera je *natanko* to, kar želimo za prikaz »pasti točnosti« – gre za odličen prikaz neuravnoteženega razvrščanja.

- Točnost ≈ 0.952
  → Približno 95.2% ur v naši razdelitvi je »brez prekoračitve«.
  Model DummyClassifier napoveduje *le* večinski razred (0), zato je glede na točnost videti »dober«.
- Preciznost = 0.0, priklic = 0.0, F1 = 0.0 → Model nikoli ne napove prekoračitve (razred 1).
  Torej:
  - Zazna 0 prekoračitev → priklic = 0

  - Ustvari 0 pozitivnih napovedi → preciznost je opredeljena kot 0 (pri zero_division = 0)
  - Zato je F1 enak 0

- PR-AUC ≈ 0.0481
  To je tukaj najbolj informativno število. Pri takšnem modelu DummyClassifier (ki vedno napoveduje isto) je PR-AUC navadno blizu deležu pozitivnega razreda. Zato je delež prekoračitev verjetno približno 4.8% (vsaj v tej ocenjevalni razdelitvi).
  To je skladno s 95% točnostjo.

[12]
✓ 0 s

```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 logistične regresije se mora učiti iz podatkov, zahtevamo pa tudi, da dobro posplošuje na še nevidene podatke z istega problemskega področja. Zato podatkovno množico običajno razdelimo na učno in testno množico. Ker je izvirna podatkovna množica časovna vrsta, bomo za preprečevanje uhajanja podatkov kot učno množico uporabili prvih 70% podatkov, kot testno množico pa zadnjih 30% (kronološka razdelitev). Tako posnemamo stvaren položaj: **učenje na preteklosti in vrednotenje na prihodnosti**. Opomba o neuravnoteženosti: po uporabi praga (mejna vrednost EU za NO_2) so lahko dogodki prekoračitve redki. Zato se zanašamo na **preciznost/priklic**, **PR-AUC** in matrike zmede, ne le na točnost. Na splošno želimo, da model dobro deluje tako

na učni kot na testni množici. Za prikaz odločitvenih mej izberemo **dve značilki** (uporabimo lahko kateri koli informativni značilki).

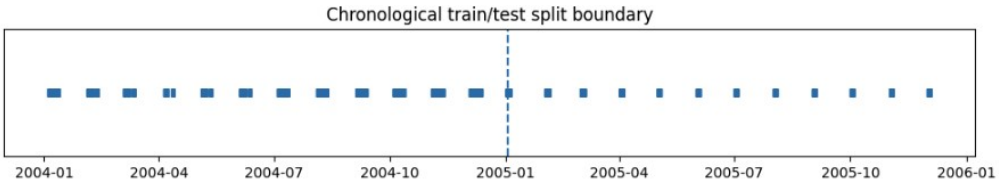
Nato naučimo **logistično regresijo** s standardizacijo in prikažemo:

- raztreseni diagram točk (obarvanih glede na razred)
- **linearno** odločitveno mejo
- izolinije verjetnosti (neobvezno)

```
[13]
✓ 0s
```

```python
# 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
```



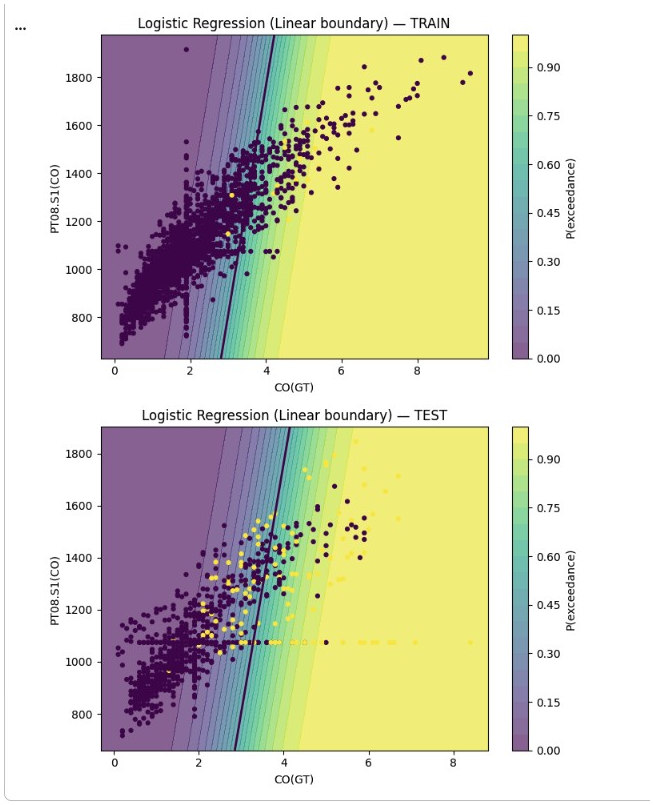
[14]
✓ 0s

```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")
```



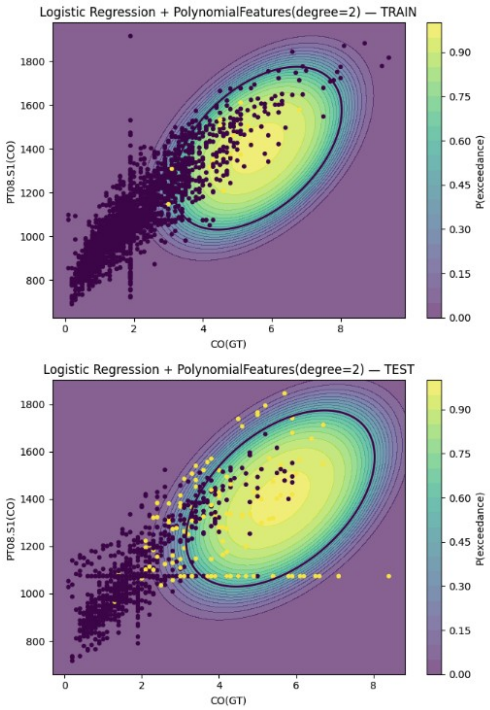
Prikaz linearne odločitvene meje kaže, kako logistična regresija razdeli razreda z ravno črto pri verjetnostnem pragu (p = 0.5), pri čemer razkrije tako ločitev razredov kot zaupanje modela na različnih območjih prostora značilk.

Logistična regresija je **linearna glede na svoje parametre**, toda s preoblikovanjem vhodnih značilk (npr. z dodajanjem kvadratnih in interakcijskih členov) lahko v izvirnem prostoru dobimo **nelinearno odločitveno mejo**. Za dodatno oblikovanje značilk uporabimo PolynomialFeatures (degree=2).

```
[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
```



Prikaz nelinearne odločitvene meje kaže ukrivljeno ločnico, ustvarjeno z oblikovanjem značilk. S tem se poveča prilagodljivost modela, hkrati pa tudi tveganje prenaučenja pri šumnih in neuravnoteženih okoljskih podatkih. Primerjava uspešnosti modelov na obeh množicah pokaže, da linearna logistična regresija presega nelinearni model (degree = 2). To nakazuje, da zadošča preprosta linearna meja, medtem ko postane nelinearni model pretirano zapleten in se začne prilagajati šumu v podatkih.

Ključna koncepta strojnega učenja sta prenaučenje in podprileganje. V nadzorovanem poskusu bomo ta pojava pojasnili z uravnavanjem zapletenosti modela s polinomsko stopnjo (zapletenostjo značilk) in močjo regularizacije C pri logistični regresiji. Natančneje, za prikaz podprileganja bomo onemogočili uteževanje razredov v logistični regresiji. Z opazovanjem uspešnosti na učni in testni množici je mogoče prepoznati posebnosti obeh pojavov.

[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
```

Ta primer podprileganja smo dobili z namerno odstranitvijo uteževanja razredov [16]. Brez tega popravka se model logistične regresije predvsem nauči napovedovati večinski razred, dogodke prekoračitve NO_2 pa v celoti spregleda. Visoka točnost, opažena na testni in učni množici, je posledica neuravnoteženosti razredov, medtem ko so priklic, mera F1 in PR-AUC na obeh množicah nizki, kar kaže, da model ne zazna redkih, vendar pomembnih primerov. To poudarja, kako se lahko model zdi uspešen, če ga presojamo samo po točnosti, čeprav se ne nauči smiselnega ali praktično uporabnega odločitvenega pravila.

```
[27]
✓ 1s
```

```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
```

Pri prenaučenju postane model pretirano prilagodljiv in se preveč prilagodi učnim podatkom. Zato zazna večino dogodkov prekoračitve, vendar sproži veliko lažnih alarmov, kar povzroči nizko preciznost in nizko mero F1 na učni množici. Ko isti model uporabimo na novih podatkih, se takšno vedenje ne prenese: priklic se zmanjša, preciznost se spremeni, splošna uspešnost pa postane manj zanesljiva. Jasna razlika med rezultati na učni in testni množici kaže, da se je model naučil vzorcev, ki se ne posplošujejo dobro, kar je značilen znak prenaučenja.

Praktična vaja: zaznavanje habitata Spruce/Fir z logistično regresijo

Podatkovna množica za to vajo: uporabimo podatkovno množico **Forest Cover Type** (pogosto imenovano *Covertype*). Vsebuje **kartografske in okoljske spremenljivke** (nadmorska višina, naklon, razdalje do vodnih virov, osenčenost pobočja, območja divjine, vrste tal …), oznaka pa predstavlja **tip gozdnega pokrova**.

Z okoljskega vidika je to značilna naloga:

- **Napovedovanje razredov habitata/pokrova iz opisnih spremenljivk terena in okolja.**

Izvirna podatkovna množica vsebuje 7 vrst pokrova (večrazredni problem). Da vaja ostane usklajena z binarno logistično regresijo in da so metrike ter prikazi lažje razložljivi, jo preoblikujemo v binarno nalogo:

- **pozitivni razred (1):** Cover Type 1 = *Spruce/Fir*
- **negativni razred (0):** vse druge vrste pokrova

To preoblikovanje odraža stvaren okoljski primer: zaznavanje določene vrste habitata, ki nas zanima, v primerjavi z vsemi drugimi vrstami pokrovnosti tal. Podatkovno množico lahko pridobimo neposredno s funkcijo fetch_covtype() iz knjižnice scikit-learn.

Orodja in okolje

- Programski jezik: Python (Google Colab)
- Knjižnice: numpy, pandas, scikit-learn, matplotlib
- Dostop do podatkovne množice: sklearn.datasets.fetch_covtype
- Način dela: samostojno delo študenta

Naloga opisa značilk

Študenti morajo pregledati izvirno dokumentacijo podatkovne množice in pripraviti preglednico z opisi značilk, ki vsebuje naslednje podatke:

- Ime značilke
- Vrsta značilke (zvezna ali kategorična / kodirana po postopku one-hot)
- Enote (če so relevantne)
- Fizikalni ali okoljski pomen

Podatkovna množica vsebuje 10 zveznih okoljskih spremenljivk in 44 kategoričnih spremenljivk (4 kazalnike območij divjine in 40 kazalnikov vrst tal; vsi so kodirani po postopku one-hot).

Naloge študenta (korak za korakom)

1. Naložite podatke iz knjižnice scikit-learn s funkcijo fetch_covtype()

2. Podatkovna množica je velika, zato iz nje z uporabo stratificiranega vzorčenja izberite podvzorec (N=50000 zapisov)
3. Uvozite vse knjižnice (enake kot v primeru uporabe iz predavanja) in zagotovite ponovljivost poskusa z nastavitvijo fiksnega naključnega semena
4. Večrazredni problem pretvorite v problem binarnega razvrščanja
    a. Razred z oznako 1 = Spruce/Fir (pozitivni razred, ki nas zanima)
    b. Razred z oznako 0 = vse druge vrste (negativni razred)

    Spruce/Fir obravnavajte kot pozitivni razred, saj je glavni cilj zaznati habitat, ki nas zanima!
5. Preučite in poročajte o ravnovesju med pozitivnim in negativnim razredom
6. Pojasnite, zakaj je binarno preoblikovanje realistično pri spremljanju okolja
7. Preverite, ali podatki vsebujejo manjkajoče vrednosti
    a. Če je manjkajočih vrednosti malo → jih odstranite
    b. Če jih je veliko → manjkajoče vrednosti nadomestite (s povprečjem ali modusom, odvisno od vrste značilke)
    c. Če jih ni, izrecno navedite »ukrepanje ni potrebno«.
8. Preverite podvojene zapise (neobvezno) in osnovne osamelce (vsaj pri zveznih spremenljivkah)
9. Ustvarite učno in testno množico (70/30) z enako porazdelitvijo razredov in z uporabo stratify = y ter pojasnite vlogo vsake množice
10. Preglejte razpone in merske lestvice prvih 10 značilk ter ugotovite, kaj je pri logistični regresiji potrebno kot začetni korak priprave podatkov.→ Pomembno: na kateri podatkovni množici je treba to izvesti?
    a. Uporabite Pipeline, ki vključuje StandardScaler.
    b. Covertype vsebuje veliko binarnih značilk 0/1, kodiranih po postopku one-hot; varneje je uporabiti StandardScaler(with_mean=False).
    c. Pojasnite, zakaj je skaliranje pomembno za LR (regularizacija je odvisna od lestvice značilk)
11. Naučite izhodiščni model logistične regresije.→ Upoštevajte, da regularizacija uravnava zapletenost: manjši C = močnejša regularizacija → preprostejši model (lahko pride do podprileganja); večji C = šibkejša regularizacija → prilagodljivejši

model (lahko pride do prenaučenja, zlasti pri oblikovanih značilkah). Logistična regresija kot izhod poda verjetnosti; privzeti prag 0.5 je poljuben.

12. Poročajte o metrikah vrednotenja na učni in testni množici (katere metrike ste izbrali in zakaj)
13. Prikažite matriko zmede na testni množici in pojasnite:
    - Pojasnite pomen TP, TN, FP in FN v tem okoljskem okviru.
    - FP → model napove »Spruce/Fir«, vendar je na lokaciji neka druga vrsta pokrova → z okoljskega vidika tvegano → napačna karta habitatov → napačne odločitve o upravljanju/varovanju
    - FN → model napove »drugo«, vendar je na lokaciji dejansko »Spruce/Fir« → z okoljskega vidika škodljivo, vendar manj nevarno kot FP → habitat je nezadostno zaščiten oziroma podcenjen
    - To kaže, kako strojno učenje (ML) podpira odločanje na področju okolja.
14. Primerjajte metrike iz 9. koraka in razložite kompromise med merami vrednotenja.
    a. Pojasnite, za katero vrsto problema gre in zakaj ga je pomembno zaznati; obravnavajte tudi zaznavanje FP in FN glede na strošek spregledanega habitata v primerjavi z lažnim alarmom.
    b. Gre za problem **zaznavanja habitata** z asimetričnimi stroški, zato spreminjanje praga vpliva na FP/FN.
15. Pojasnite preciznost, priklic in PR-AUC posebej za nalogo zaznavanja habitata Spruce/Fir.
16. Povečajte prilagodljivost modela s polinomsko razširitvijo značilk (stopnje 1, 2 in 4), ki jo uporabite samo za zvezne značilke, ter spreminjajte moč regularizacije (C = 0.1, 1.0, 50). Pojasnite pričakovane učinke na vedenje modela.
    a. Da se Colab ne sesuje, uporabite polinomske značilke le za **10 zveznih spremenljivk**.
17. Za vse nastavitve modela prikažite mero F1 na učni in testni množici ter razložite opažene vzorce.
18. Na podlagi rezultatov prejšnjega koraka pojasnite koncepta podprileganja in prenaučenja.

19. Določite eno nastavitev s podprileganjem in eno s prenaučenjem ter svojo izbiro utemeljite z numeričnimi rezultati.
20. Prikažite odločitvene meje v dvorazsežnem prostoru (2D), pri čemer za zmanjšanje razsežnosti uporabite PCA (samo za prikaz), ter opišite, kako se meje spreminjajo z večanjem zapletenosti modela.

Kaj morate oddati – poročilo o vaji

- Oblika: Google Colab s celicami Markdown in celicami s kodo
- Vsebovati mora:
    a. Uvod (3–5 stavkov)
        I. Kaj je bil problem? (izvirni in preoblikovani)
        II. Zakaj je to pomembno za okoljske vede?
        III. Opredelite pozitivni razred in pojasnite asimetrični strošek v tem primeru.
    b. Opis podatkov
        I. Število primerov in število značilk (razvrščenih po vrstah)
        II. Ciljna spremenljivka in njena porazdelitev
        III. Obravnava manjkajočih podatkov (napišite, kaj ste storili)
        IV. Odločitev o podvzorčenju
    c. Metode
        I. Kateri modeli so bili naučeni in kako (LogReg + parametri)
        II. Razdelitev na učno/testno množico (navedite razmerje)
        III. Morebitno skaliranje ter podatkovna množica, na kateri ste ga izvedli
    d. Rezultati
        I. Predstavite točnost, preciznost, priklic, oceno F1, ROC-AUC in PR-AUC (v obliki preglednice)
        II. Prikažite matriko zmede z razlago
        III. Navedite, kateri model bi izbrali in zakaj (ne zgolj zato, ker »ima višjo točnost«, temveč morda zato, ker »ima višji priklic pri pozitivnem razredu«)
    e. Razprava

I. Kakšna so pri tem primeru tveganja napačnega razvrščanja v okoljskih aplikacijah?
II. Kakšna je vloga skaliranja?
III. Kaj bi storili v naslednjem koraku, če bi se to uporabljalo za spremljanje okolja? (npr. prilagajanje praga oziroma kompromis med preciznostjo in priklicem, pomembnost značilk oziroma interpretabilnost)

f. Sklep
I. V nekaj povedih povzemite, kaj ste se pri tej vaji naučili z uporabo resničnih okoljskih podatkov – omenite posploševanje, izbiro metrik in kompromise.

Ocenjevanje

- **Kviz** (10 vprašanj): zasnovan bo za preverjanje konceptualnega razumevanja in uporabnega sklepanja, pridobljenega na predavanju in pri praktični vaji o tehnikah razvrščanja v okoljskih znanostih. Preverjal bo, ali študenti razumejo nadzorovano učenje, odločitvene meje (linearne in nelinearne), osnovne metrike vrednotenja razvrščanja (točnost, preciznost, priklic, oceno F1), vlogo razdelitve na učno in testno množico, pojava podprileganja in prenaučenja ter pomen vizualizacije podatkov.

- **Praktična naloga razvrščanja:** napovedovanje prekoračitve NO_2 – ta samostojna praktična vaja študentom omogoča praktično delo s problemom razvrščanja, ki izhaja iz scenarija spremljanja okolja, obravnavanega na predavanju. Z uporabo podatkovne množice o kakovosti zraka študenti naučijo in ovrednotijo izhodiščni linearni model (logistično regresijo), da ugotovijo, ali ravni dušikovega dioksida (NO_2) presegajo določen varnostni prag. Naloga je zasnovana za preverjanje sposobnosti študentov, da izvedejo celoten potek strojnega učenja, vključno s pripravo podatkov, obravnavo značilk, učenjem modela in vrednotenjem. Poseben poudarek je na razlagi lažno pozitivnih in lažno negativnih rezultatov v okviru podpore odločanju o kakovosti zraka ter na prikazu, kako lahko modeli razvrščanja podpirajo spremljanje okolja in ocenjevanje tveganj.

- **Praktična naloga razvrščanja: zaznavanje habitata (Spruce/Fir)** – ta samostojna praktična vaja študentom omogoča praktično uporabo modelov razvrščanja pri okoljskem primeru, predstavljenem na predavanju. Z uporabo

podatkovne množice o pokrovnosti tal z geografskimi in okoljskimi spremenljivkami študenti zgradijo in ovrednotijo izhodiščni linearni klasifikator (logistično regresijo) za zaznavanje prisotnosti določene vrste habitata (smrekov/jelov gozd, Spruce/Fir) v primerjavi z vsemi drugimi vrstami pokrovnosti tal. Naloga je zasnovana za preverjanje sposobnosti študentov, da izvedejo celoten potek strojnega učenja, vključno s pripravo podatkov, obravnavo značilk, učenjem modela in vrednotenjem. Poseben poudarek je na razlagi lažno pozitivnih in lažno negativnih rezultatov v okviru podpore okoljskemu odločanju ter na prikazu, kako lahko modeli razvrščanja pomagajo pri kartiranju habitatov, načrtovanju ohranjanja narave in analizi rabe zemljišč.

- **Kratko pisno poročilo**: kratko pisno poročilo je zadnji korak praktične vaje zaznavanja habitata. Njegov namen je študentom omogočiti, da povzamejo, razložijo ter jasno sporočijo rezultate, pridobljene z modeli razvrščanja pri problemu zaznavanja habitata smrekovega/jelovega gozda (Spruce/Fir). V poročilu naj študenti izkažejo sposobnost:
  - jasno pojasniti korake svojega poteka razvrščanja, od priprave podatkov do vrednotenja modela,
  - razložiti metrike razvrščanja (točnost, preciznost, priklic, oceno F1 in matriko zmede) v okviru zaznavanja habitatov in okoljskega odločanja ter
  - razmisliti o izbiri modela (linearnega ali nelinearnega) ter razpravljati o tem, kako kompleksnost modela vpliva na uspešnost in kaj to pomeni za praktične uporabe, kot so kartiranje habitatov, načrtovanje ohranjanja narave in analiza rabe zemljišč.

Merila ocenjevanja

ZA KVIZ

- Kviz bo preverjal poznavanje dejstev ter konceptualno in temeljno teoretično razumevanje razvrščanja pri strojnem učenju, vključno z naslednjim:
  - paradigma nadzorovanega učenja,
  - razlika med linearnimi in nelinearnimi odločitvenimi mejami,
  - vloga razdelitve na učno/validacijsko/testno množico,
  - metrike razvrščanja in okoliščine njihove uporabe,
  - primeri problemov okoljskega razvrščanja (prekoračitev NO_2 in zaznavanje habitata)

Prag uspešnosti: 50 % pravilnih odgovorov.

ZA PRAKTIČNO NALOGO RAZVRŠČANJA

- Priprava in predobdelava podatkov
  - Pravilno nalaganje in pregled podatkovne množice (oblika, ciljna spremenljivka, manjkajoče vrednosti).
  - Ustrezna obravnava manjkajočih podatkov (odstranjevanje ali nadomeščanje vrednosti) ter po potrebi skaliranje/normalizacija značilk.
  - Pregleden in ponovljiv potek predobdelave (koda je urejena, koraki pa jasni).

- Izvedba modelov
  - Pravilna nastavitev in učenje najmanj dveh modelov razvrščanja:
    - enega izhodiščnega (npr. logistične regresije),
    - drugega nelinearnega modela
  - Ustrezna uporaba razdelitve na učno in testno množico (ter validacijsko množico, če je uporabljena).
  - Koda je urejena, parametri pa so izrecno navedeni.

- Vrednotenje in diagnostika modelov
  - Pravilen izračun in poročanje o metrikah razvrščanja:
    - točnost
    - preciznost
    - priklic
    - ocena F1
    - matrika zmede
    - Smiselna primerjava modelov (npr. »Logistična regresija je v primerjavi z nelinearnim modelom, pridobljenim z oblikovanjem značilk, izboljšala priklic pri pozitivnem razredu«).
    - Zavedanje neuravnoteženosti razredov in njenega vpliva na točnost.

- Izbira modela in utemeljitev
  - Jasna utemeljitev izbire prednostnega modela (ne le »višja točnost«, temveč, kadar je to pomembno, »boljši priklic pri manjšinskem oziroma okoljsko kritičnem razredu«).
  - Smiselna razlaga sposobnosti posploševanja modela na podlagi rezultatov na testni množici

- Vizualizacija in predstavitev rezultatov
  - Matrika zmede je grafično prikazana, označena in razložena.
  - Izbira najboljšega modela na podlagi dejanskih podatkov iz preglednice ali grafa
  - Grafični prikazi so berljivi in podpirajo argumentacijo.

- Povezava z okoljskim kontekstom
  - Študent izkaže razumevanje napačnega razvrščanja pri zaznavanju habitatov in se zaveda, da je napačna razvrstitev območij brez habitata smrekovega/jelovega gozda (Spruce/Fir) kot območij s tem habitatom na splošno resnejša napaka kot spregled resničnega primera habitata.
  - Študent zna pojasniti, kako bi tak klasifikator lahko podprl spremljanje okolja, načrtovanje ohranjanja narave in upravljanje rabe zemljišč, saj bi pomagal prepoznavati in spremljati razširjenost habitatov smrekovega/jelovega gozda (Spruce/Fir), določati prednostna območja za zaščito ali terenske preglede ter podpirati okoljsko odločanje na podlagi podatkov.

ZA KRATKO PISNO POROČILO

- Jasnost in zgradba poročila
  - Poročilo je smiselno strukturirano (Uvod → Podatki → Metode → Rezultati → Razprava → Sklep).
  - Koraki poteka dela so razvidni v zvezku
  - Jezik je natančen in jedrnat.

- Razlaga rezultatov in razprava
  - Rezultati razvrščanja (točnost, preciznost, priklic, ocena F1, matrika zmede) so pravilno navedeni in smiselno razloženi.
  - Študent iz matrike zmede razbere napake za posamezne razrede.
  - Študent razmisli o tem, zakaj je en model dosegel boljše rezultate od drugega
  - Razprava kaže zavedanje o prenaučenju in podprileganju ter o vlogi validacije.

- Povezava z ekološkim in okoljskim kontekstom
  - Ugotovitve so povezane z njihovim resničnim okoljskim pomenom (npr. »višji priklic pri razredu Spruce/Fir je pomembnejši od skupne točnosti«).
  - Študent izkaže razumevanje, kako lahko razvrščanje na podlagi podatkov prispeva k spremljanju okolja in informiranemu odločanju.

- Kakovost vizualne predstavitve
  - Slike (matrika zmede, metrike vrednotenja, primerjava modelov) so jasno označene.
  - Na grafične prikaze se besedilo sklicuje (»kot prikazuje slika 1 …«).
  - Grafični prikazi pomagajo pojasniti vedenje modela in niso zgolj okras poročila.

- Sporočilnost in slog
  - Besedilo je jasno, objektivno in strokovno.
  - Terminologija strojnega učenja in okoljska terminologija sta uporabljeni pravilno (razred, oznaka, značilka, preciznost, priklic, pozitivni razred).
  - Študent upošteva omejitve (majhna podatkovna množica, neuravnoteženost in odsotnost prostorske komponente), izzive visokorazsežnih podatkov ter potrebo po podvzorčenju velikih podatkovnih množic pri delu v brezplačnem okolju Google Colab.

KVIZ (10 vprašanj)

1. Nadzorovano učenje

Katera od naslednjih trditev najbolje opisuje nadzorovano učenje pri analizi okoljskih podatkov?

A. Model odkriva vzorce v podatkih brez vnaprej določenih oznak
B. Model se nauči preslikave od vhodnih značilk do znanih ciljnih oznak
C. Model samodejno združuje podobna okoljska opazovanja
D. Model optimizira pravila odločanja brez uporabe učnih podatkov

2. Binarno preoblikovanje

Zakaj je bil prvotni večrazredni problem razvrščanja vrst gozdnega pokrova preoblikovan v nalogo binarnega razvrščanja?

A. Logistične regresije ni mogoče uporabiti pri večrazrednih problemih
B. Da bi poenostavili razlago metrik in odločitvenih mej
C. Podatkovna množica vsebuje le dve smiselni vrsti pokrovnosti
D. Da bi se izognili skaliranju značilk

3. Opredelitev pozitivnega razreda (zaznavanje habitata)

Kaj pri nalogi zaznavanja habitata smrekovega/jelovega gozda (Spruce/Fir) predstavlja pozitivni razred (1)?

A. Katero koli območje, poraslo z gozdom
B. Območja z visoko nadmorsko višino
C. Lokacije, ki jih pokriva smrekov/jelov gozd (Spruce/Fir)
D. Lokacije z minimalnim vplivom človeka

4. Lažno pozitivni rezultati pri zaznavanju NO_2

Kaj pri nalogi razvrščanja prekoračitev NO_2 predstavlja lažno pozitiven rezultat?

A. Pravilna prepoznava nevarne kakovosti zraka
B. Napoved varne kakovosti zraka, ko NO_2 presega prag
C. Napoved prekoračitve NO_2, ko je kakovost zraka v resnici varna
D. Neuspešno učenje modela

5. Resnost napačnega razvrščanja (zaznavanje habitata)

Katera napačna razvrstitev pri nalogi zaznavanja habitata običajno pomeni večje okoljsko tveganje?

A. Lažno negativna napoved
B. Lažno pozitivna napoved
C. Pravilno negativna napoved
D. Pravilno pozitivna napoved

6. Izbira metrike

Katera metrika je najprimernejša, če je glavni cilj zaznati čim več resničnih habitatov smrekovega/jelovega gozda (Spruce/Fir)?

A. Točnost

B. Preciznost
C. Priklic
D. Specifičnost

7. Preciznost v primerjavi s priklicem

V katerem primeru bi bila preciznost pomembnejša od priklica?

A. Kadar je habitat povezan z zelo visokimi stroški
B. Kadar lažni alarmi vodijo v drage terenske preglede
C. Kadar je podatkovna množica popolnoma uravnotežena
D. Pri vizualizaciji odločitvenih mej

8. Razdelitev na učno in testno množico

Kaj je glavni namen razdelitve podatkov na učno in testno množico?

A. Zmanjšati velikost podatkovne množice
B. Prilagoditi hiperparametre
C. Oceniti posploševanje na predhodno nevidene podatke
D. Izboljšati vizualizacijo

9. Podprileganje v primerjavi s prenaučenjem

Model dosega zelo dobre rezultate na učnih podatkih, vendar slabe na testnih podatkih. To je primer:

A. podprileganja
B. prenaučenja
C. neuravnoteženosti razredov
D. naključne napake

10. Vizualizacija odločitvene meje

Zakaj so bile pri praktičnih vajah odločitvene meje prikazane v dveh razsežnostih z uporabo PCA?

A. Da bi izboljšali točnost modela
B. Da bi zmanjšali prenaučenje
C. Da bi omogočili interpretacijo vedenja visokorazsežnega modela

D. Da bi nadomestili metrike vrednotenja

Predlagane uteži

- Kviz: 20 %
- Praktična naloga razvrščanja (zvezek/koda): 50 %
- Kratko pisno poročilo: 30 %

Viri/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.