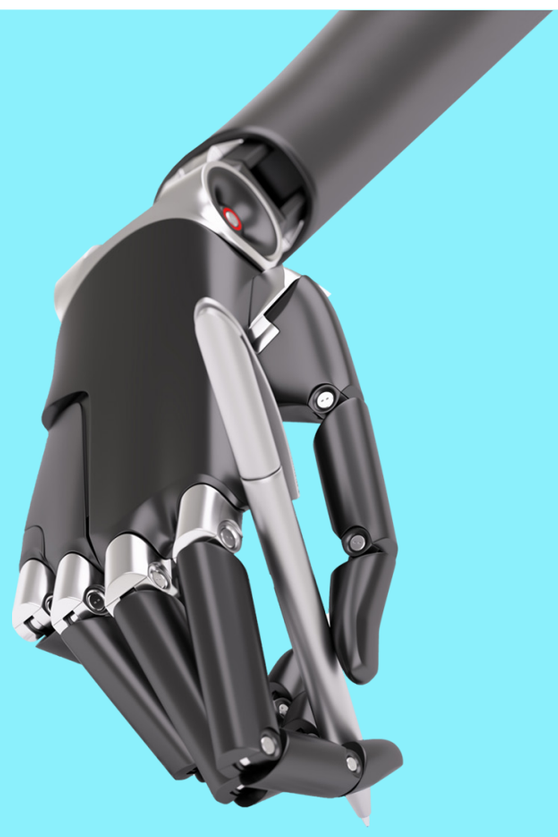
چالش ورودی بوت‌کمپ
ماشین لرنینگ
MACHINE LEARNING BOOTCAMP



چالش ورودی بوت‌کمپ ماشین لرنینگ

از اینکه علاقه مند به شرکت در بوت کمپ ماشین لرنینگ رهنماکالج هستید، خوشحالیم. در فرآیند ثبت نام علاوه بر سوالاتی که در مورد رزومه و انگیزه شما پرسیده شده، لازم است به سوالات زیر پاسخ دهید و پاسخ تان را در فرم ثبت نام برای ما بارگذاری کنید.

این تمرین‌ها برای این طراحی شده‌اند که ببینیم شما چگونه فکر می‌کنید و چگونه یاد می‌گیرید. انتظار نداریم از قبل دانشی در زمینهٔ یادگیری ماشین داشته باشید. در هر تمرین، یک الگوریتم ساده در اختیارتان قرار می‌گیرد تا آن را از ابتدا یاد بگیرید و روی یک مجموعه‌دادهٔ واقعی پیاده‌سازی کنید. برای ما، نحوهٔ برخورد شما با مسئله و فرایند یادگیری‌تان بسیار مهم‌تر از رسیدن به یک عدد یا پاسخ کاملاً درست است.

دو تمرین در نظر گرفته شده است:

- **تسک A نزدیک‌ترین همسایه‌ها (Nearest Neighbours)**
  این تمرین به ریاضیات پیشرفته نیاز ندارد و برای بیشتر متقاضیان گزینهٔ مناسب‌تری است.
- **تسک B رگرسیون خطی با گرادیان کاهشی (Linear Regression with Gradient Descent)**
  این تمرین کمی ریاضی‌محورتر است. اگر از مباحث ریاضی لذت می‌برید و می‌خواهید توانایی خود را در این زمینه نشان دهید، می‌توانید این تسک را به‌جای تسک A یا علاوه‌بر آن انجام دهید.

اگر در انتخاب مردد هستید، پیشنهاد ما انجام تسک A است.

می‌توانید از دستیارهای هوش مصنوعی مانند ChatGPT، Claude یا Copilot استفاده کنید؛ در واقع، ما شما را به استفاده از آن‌ها تشویق می‌کنیم. تنها انتظار ما این است که نشان دهید چگونه از این ابزارها استفاده کرده‌اید (به بخش «An AI-use note» مراجعه کنید). همچنین لازم است هر چیزی را که تحویل می‌دهید، به‌خوبی درک کرده باشید؛ زیرا در یک گفت‌وگوی کوتاه از شما خواسته می‌شود آن را توضیح دهید و گسترش دهید.

انجام دادن هر دو تسک به عنوان امتیاز محسوب می‌شود.

زمان تقریبی موردنیاز: حدود ۴ تا ۶ ساعت که می‌تواند در چند روز تقسیم شود.
ابزار موردنیاز: فقط Python3 و یک ویرایشگر کد.
زبان: گزارش یادگیری و یادداشت‌های خود را می‌توانید به فارسی بنویسید، اما کد، نام متغیرها و توضیحات داخل کد را به انگلیسی نگه دارید. از آنجا که منابع و مستندات این حوزه عمدتاً به زبان انگلیسی هستند، راحت بودن با خواندن متون انگلیسی می‌تواند کمک بزرگی باشد.

Task A — Build a Nearest Neighbours Classifier from Scratch

What you'll build

A program that, given a new example, predicts which class it belongs to by looking at the most similar examples it has already seen. This is the k-Nearest Neighbours (kNN) algorithm, and it's one of the simplest ideas in machine learning: similar things tend to have similar labels.

You will implement it yourself using only `numpy` (and the standard library). Do not use a ready-made kNN from a library such as scikit-learn for your solution — that's the part we want to see you build.

The dataset

We'll use the Wine dataset: 178 wines, each described by 13 chemical measurements (alcohol, magnesium, colour intensity, and so on). Each wine belongs to one of 3 cultivars (classes `0`, `1`, `2`). Your job is to predict the class from the 13 measurements.

It ships inside scikit-learn, so there's nothing to download. Use this starter code to load it and split it into a part to learn from and a part to test on:

```
import numpy as np
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split

X, y = load_wine(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# From this point onward, implement your OWN k-Nearest Neighbours algorithm.
# Do NOT use sklearn's KNeighborsClassifier in your solution.
```

Dataset reference (description of the 13 features): HERE

What to do

1. Implement kNN from scratch. For a new wine, find the `k` training wines most similar to it (you decide how to measure "similar"), and predict the most common class among them.
2. Run your classifier on the test set and report **how accurate it is** (what fraction of test wines it labels correctly).
3. Try at least three different values of `k` and observe what happens to the accuracy.

What to hand in

1. **Your code** (a `.py` file or notebook), runnable, with brief comments.
2. **A short results section:** your accuracy, and a small table or plot of accuracy versus `k`.
3. **A one-page learning log**answering, honestly and in your own words:
   - What did you already understand, and what was genuinely new to you?
   - What confused you or didn't work the first time, and how did you get unstuck?
   - How did you decide what `k` to use, and why do you think it mattered?
   - One question you still have, or one thing you'd improve with another day.
4. **An AI-use note** (see below).

Task B — Build Linear Regression from Scratch with Gradient Descent

What you'll build

A program that predicts a number (not a category) by fitting a straight-line relationship to data, and that learns the best line by repeatedly nudging it in the direction that reduces its error. The learning method is called gradient descent, and it's the same basic engine that trains modern neural networks — so this is a small version of something very real.

Implement it yourself using only `numpy`. Do not use scikit-learn's `LinearRegression` or `SGDRegressor` for your solution.

The dataset

The California Housing dataset: about 20,600 districts in California, each described by 8 numbers (median income, house age, average rooms, population, etc.). The target you predict is the median house value of the district (in units of \$100,000).

It also ships inside scikit-learn. Starter code:

```
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

data = fetch_california_housing()
X, y = data.data, data.target      # X: 20640 rows x 8 features,  y: median house value ($100k)
print(data.feature_names)          # ['MedInc', 'HouseAge', 'AveRooms', ...]

# TIP: to get a first working version you can plot, start with ONE feature —
# median income is the strongest single predictor:
#   X = X[:, [0]]   # column 0 = MedInc
# Once that works, come back and use all 8 features.

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# From here on, write your OWN linear regression trained with gradient descent.
# Do NOT use sklearn's LinearRegression / SGDRegressor for your solution.
```

Dataset reference (description of the 8 features): HERE

What to do

1. Start with a single feature (median income) so you have something you can plot: the data points and the line your model learns.
2. Implement gradient descent: define how wrong the line is (an error/loss), and repeatedly adjust the line to make that error smaller.
3. **Record the error at each step** and plot it over time — you should be able to watch it go down.
4. Then extend your model to use all 8 features and report how well it predicts on the test set.

What to hand in

1. **Your code** (a `.py` file or notebook), runnable, with brief comments.
2. **A plot of the error over training steps** (the "loss curve"), plus, for the single-feature version, a plot of the data with your fitted line.
3. **A short results section:** how well your model predicts on the test set, and what changed when you went from 1 feature to all 8.
4. **A one-page learning log**answering, honestly and in your own words:
   - What did you already understand, and what was genuinely new?
   - What did you try that didn't work, and how did you fix it? (For example: did

training ever behave strangely? What did you do?)
  - How did you choose how big each adjustment step is, and what happened when you changed it?
  - One question you still have, or one thing you'd improve with another day.
5. **An AI-use note** (see below).

Using AI (please read — this part matters to us)
You're welcome to use AI assistants. We use them every day, and knowing how to work with them is part of what this bootcamp teaches. But we want to

see your thinking, not just an answer, so:
- Keep and submit your chat transcript(s) (a link or an exported file is fine).
- Add a few sentences: **Where was the AI wrong, vague, or did something you had to fix or push back on?** Point to at least one concrete moment where you steered, corrected, or chose not to follow it.
A submission that you can't explain afterwards tells us less than a rough one you fully understand. If you hand in something you didn't read, it will become clear very quickly when we talk it through.

What we're looking for (so you know)
- **You can learn something new from a resource and apply it.** This is the single most important thing.
- **Honesty about what was hard.** "This confused me, here's how I worked it out" is a *strong* answer, not a weak one.
- **Curiosity.** Noticing something odd and digging into *why* counts for a lot.
- **Clear, working Python.** It doesn't need to be elegant; it needs to run and make sense.

We are explicitly not looking for prior ML knowledge, a fancy university, or a polished portfolio. Show us how you think.
Curated resources
You don't need anything beyond these, but you're free to use whatever helps.

For Task A (Nearest Neighbours)
- Video, gentle and intuitive — StatQuest, "K-nearest neighbors, Clearly Explained": HERE
- Reference / overview — Wikipedia, "k-nearest neighbors algorithm": HERE
For Task B (Linear Regression + Gradient Descent)
- Video, step by step — StatQuest, "Gradient Descent, Step-by-Step": HERE - For deeper visual intuition on the math (optional) — 3Blue1Brown, the *Essence of Calculus* and *Neural Networks* series: HERE

General
- StatQuest video index (a well-organised hub, basics > advanced): HERE
- After you finish, it's worth comparing your result against the library version — scikit-learn's `KNeighborsClassifier` (Task A) or `LinearRegression` (Task B) — to sanity-check your numbers. Use these only to check, not to build your solution.

هدف نهایی تمرین:

با این تمرین می خواهیم شما نه تنها در زمینه الگوریتم ها و مدل‌های یادگیری ماشین متخصص شوید، بلکه مهارت‌های مهمی مانند کار تیمی، اعتمادسازی، و رهبری مؤثر را نیز تقویت کنید، چون موفقیت یک پروژه فنی، فقط به کدنویسی وابسته نیست.

دقت کنید!

- لطفا در هر سوال که با کمبود اطلاعات مواجه بودید، فرضیاتی را در نظر بگیرید و این فرضیات را در آخر پاسخ تان ذکر کنید.
- پاسخ های خود را باید در قالب یک فایل PDF ارسال کنید که نام آن ترکیبی از نام و نام خانوادگی شما به طور مثال، Ali.Akbari باشد.
- هر چند جست وجوی فردی بخشی از مراحل حل مسئله است با این وجود نهایتا ما در ارزیابی‌ها و بررسی پاسخ‌ها مبنا را بر صداقت شما می‌گذاریم و فرض می‌کنیم به این موضوع واقف هستید که پاسخ غیرصادقانه شما، ما را در ارزیابی صحیح دچار گمراهی می‌کند و ممکن است وارد دوره‌ای شوید که متناسب با نیازتان نیست.
- با توجه به محدود بودن ظرفیت دوره، چالش‌ها به ترتیب ارسال، بررسی و ارزیابی می‌شوند و با پر شدن ظرفیت، مهلت ارسال چالش به اتمام می‌رسد.

به امید دیدارتان هستیم

رهنما کالج
با حرفه‌ای‌ها بگرد!