CareerSwami · Data Science & Machine Learning

FREE TWO-WEEK INTERNSHIP

Project 1 — Predict bike-rental demand without fooling yourself

A hundred and eighty days of synthetic data with a known rule inside it. Any model will score well. The project is whether you can split the data honestly, beat a baseline that takes one line, and say plainly what your numbers do not prove.

2 weeks	1 project	6–8 hrs
START TO SUBMIT	ONE SUBMISSION	PER WEEK

WHAT IS IN THIS BRIEF

1. The situation — the problem you are being handed
2. The four ways to fool yourself with this dataset — the cases your work must survive
3. What you are given — your data and starting files
4. What finished looks like — the two things you hand in
5. How the two weeks run — week by week
6. Before you start — rules, tools and your starter kit
7. How the work is judged — the marking scheme
8. How to submit — one link, pasted into your dashboard

Read 1 to 4 before you do anything. They take five minutes and they are what the marking scheme is built on. Sections 5 to 8 are the how — come back to them once you know what you are solving.

1 The situation

Treat this as a real client brief. The data is generated from a formula you are given, which means — unusually — you can check whether your model found the truth or just memorised the noise.

You generate a hundred and eighty sequential days with a fixed random seed, so everyone's dataset is identical and your results are reproducible. Temperature follows a seasonal curve with noise. Rain happens about a quarter of the time. Demand depends on temperature, rain and whether it is a weekday — plus noise that nothing can predict.

The split is given: days one to one hundred and twenty for training, one hundred and twenty-one to one fifty for validation, one fifty-one to one eighty for test. It is sequential rather than random, and that is deliberate — this is time-ordered data, and shuffling it would let tomorrow leak into yesterday.

The whole exercise is built around one discipline: using only the information that would actually be available at the moment of prediction. It is extremely easy to build a model here that scores beautifully and is worthless, and almost nobody notices on their own.

WHAT YOU ARE BEING ASKED TO DO

Generate the dataset reproducibly, explore it, split it in time order, and build a naive forecast that every later model has to beat — then report honestly whether anything beat it.

2 The four ways to fool yourself with this dataset

Every one of these produces a better-looking score and a worse model. Your split diagram and model notes should show you avoided each.

Shuffling time-ordered data

The split must stay sequential.

Why it invalidates the result — A random split puts day 175 in training and day 40 in test. The model has effectively seen the future, the error looks excellent, and the number is meaningless. This is the single most common error in time-series work and it is invisible in the output.

Using information from the prediction day itself

Only what is known before the day starts.

Why it invalidates the result — Same-day rain and temperature are in the file because they generated demand — but at prediction time you would not have them. Using them makes the model superb and unusable. Decide what is genuinely available and write the decision down.

Scaling or imputing before splitting

Fit any transformation on training data only.

Why it invalidates the result — Compute a mean across the whole dataset and your training step has absorbed information from the test set. The leak is small, invisible, and enough to make a comparison between two models dishonest.

No baseline

A naive forecast every model must beat.

Why it invalidates the result — An error figure on its own is not a result. Predicting yesterday's demand, or the training mean, takes one line and is often startlingly hard to beat. Without it you cannot tell whether your model learned anything at all.

The noise term is in the generator on purpose. There is a floor below which no model can go, and understanding that is the difference between tuning and superstition.

3 What you are given

The generator is specified exactly so every student's data is the same. What is yours is the honesty of the method.

The generation rule, seed 42, 180 days

```
weekday = day mod 7
temp = 15 + 10·sin(day/28) + uniform(-2,2)
rain = 1 if next seeded random < 0.25 else 0
demand = max(0, round(40 + 2·temp - 18·rain + 10·(weekday<5) + uniform(-8,8)))
```

The split and the baseline, worked, so the bar is clear

`Train`	days 1–120
`Validation`	days 121–150
`Test`	days 151–180 — untouched until the end
`Naive baseline A`	predict the training mean for every day
`Naive baseline B`	predict yesterday's demand
`Measure`	mean absolute error, on validation
`Report`	both baselines, before any model — they are the bar

Four steps the work must show

`Generate` `Explore` `Split` `Baseline`

This dataset is synthetic and no real-world forecasting claim may be made from it anywhere in your write-up. Record the generation order so results are reproducible. Use only information available at prediction time. Where you make a judgement call — what counts as available, how you handle the noise floor — write it down.

4 What finished looks like

At the end of two weeks someone should be able to run your code, get your numbers, and see exactly how you split the data.

- **A brief, the generation code and a data dictionary** — the question, who would use the answer and what they would do with it; the seeded generation script producing byte-identical data on any machine; and every column defined, including when in time it would actually become known.
- **Exploratory analysis, a split diagram and a naive forecast** — what the data looks like before any modelling, a diagram of the three time ranges with their day numbers, and at least one baseline with its

error measured on validation — the number every later model has to beat.

A naive baseline is not a formality. Report it as a headline result, and if nothing you build beats it, report that too — it is a finding, not a failure.

5 How the two weeks run

Work through it in two parts — but you submit **once**, at the end, not once per week.

PART 1 · WEEK 1

Generate it reproducibly and say what each column means

Write the generator with the seed fixed, confirm it reproduces exactly, and document every column — including the part most people skip, which is when each value would actually be known.

Produce

- A brief: the question, the user, the decision it would inform
- The seeded generation code, producing identical data on re-run
- A data dictionary including when in time each column becomes known

> Run the generator twice and show the outputs are identical. Record the generation order. Your dictionary must mark same-day rain and temperature as not available at prediction time, or justify in writing why you treat them as available.

PART 2 · WEEK 2

Look at it, split it honestly, and set the bar

Explore before modelling. Draw the split so the time order is unmistakable. Then build the simplest possible forecast and measure it properly.

Produce

- Exploratory analysis: distributions, the weekday effect, the rain effect
- A split diagram showing days 1–120, 121–150 and 151–180
- At least one naive forecast with its validation error reported

> The split must be sequential, not random — show the day ranges. Report mean absolute error on validation only, and do not touch the test set in week two; it exists for the final evaluation and spending it early is how a result becomes unreportable.

6 Before you start

- Split sequentially. Never shuffle time-ordered data, and show your day ranges so a reviewer can see you did not.

- Decide which features would genuinely be available before the day starts, and state the decision. Both answers can be defensible; silence cannot.
- Fit any scaling, encoding or imputation on training data only, then apply it to validation and test.
- Report the naive baseline as a headline number. A model error with nothing to compare it against is not a result.
- Do not evaluate on the test set in these two weeks. It is spent once, at the end of the full project.

Your tools

Python with pandas, numpy and scikit-learn, in a notebook or plain scripts. Google Colab is free and more than sufficient. No GPU, no paid tool and no cloud account is required, and no ID document is ever required.

CS_data-science-machine-learning_Starter_Kit.zip — bike_demand.csv, generate_data.py, a data dictionary template, a split diagram template, an experiment log, an evidence log, and a file index. https://careerswami.com/wp-content/uploads/careerswami-projects/CS_data-science-machine-learning_Starter_Kit.zip

7 How the work is judged

WHAT IS ASSESSED	WEIGHT
Problem and requirements — did you understand what you were solving	**20%**
The core artefact — the thing you actually made	**40%**
Validation and evidence — how you know it works	**25%**
Clarity and reproducibility — can someone else open and follow it	**15%**

Strong	Complete, correct, evidence-backed and easy to inspect.
Developing	A useful partial result, with the gaps explained.
Needs revision	Core deliverable missing, claims unverified, or the work cannot be opened.

Credit reflects demonstrated work, not visual polish. A small, honest result beats a large, unsupported one.

What counts as evidence

expected result → actual result → the metric, plot or code cell it came from → conclusion. Then say what the evidence does not establish.

Why this one is worth doing. Everyone can call fit and predict. What distinguishes someone worth hiring is that they split in time order, state what the model would actually know on the day, and publish a naive baseline that makes their result meaningful. Those three habits are also exactly what an experienced reviewer checks first — and most candidates fail all three without ever being told why they were rejected.

8 How to submit

One link, pasted into your dashboard

1. Put your work somewhere with a link — Google Drive, Figma, GitHub, Notion, Loom. Whatever fits the work.
2. **Set it so anyone with the link can view.** A locked link is the single most common reason a submission comes back.
3. Include a short README or cover page saying what to open first, and where each deliverable lives.
4. Open your dashboard, paste the link into the submit box, and add a line or two on anything you want the reviewer to know.

Name files `CS_data-science-machine-learning_YourName_v1`. Keep your own copy. A screenshot on its own is not enough for anything interactive. A video is optional — annotated screenshots or narrated slides are fine.

Submitted and heard nothing, or the link came back? Email **support@careerswami.com** with the email address you enrolled with.

Work honestly. Use fictional data and your own work. Credit anything you reused, and disclose AI assistance — including what you checked yourself. Never invent interviews, users, tests or results. Nothing here requires you to spend money, contact real organisations, or deploy anything publicly. Never upload identity documents, real customer records or credentials.

Stuck, confused, or something is broken? Write to us.

`support@careerswami.com`

Or WhatsApp **+91 7383138418**. We answer on working days. Use it for anything at all — a part of this brief that does not make sense, a starter file that will not open, a dashboard link that has stopped working, a deadline you need moved, or a submission you are not sure about.

To get a useful answer fast, spend twenty minutes narrowing the problem down first, then tell us four things: what you expected, what actually happened, what you already tried, and your one specific question. Include your track name and the email you enrolled with.

Asking is not penalised. Submitting partial work with a clearly stated blocker always scores better than a finished answer you copied, and better than a silent no-show.

Optional stretch

Optional, and only once the main work is done — cap it at an hour. Estimate the noise floor from the generator and say how close the baseline already is to it, or add a rolling-origin validation scheme. Label it clearly as a stretch.