Before the hands-on lab, the process of building a machine learning model felt like a simple, step-by-step recipe: gather data, clean it up, train the model, and evaluate. I saw the ML workflow as a long checklist that started at the beginning and finished at the end. However, working through the steps showed me that it’s actually a complex cycle of building and fixing.

The real challenge isn't just completing each step, but realizing how they all depend on each other. For example, if the model doesn't perform well during the evaluation phase, the solution is rarely just swapping out the algorithm. We often have to loop right back to the beginning to the data preparation or feature engineering. If the model struggles to learn, the problem might be that the input data is missing an important piece of information or that the numbers aren't scaled correctly. This means we must go back, create new features, re-clean the data, and then try training the model all over again. This constant movement back and forth where the test results force us to change the inputs is what makes the workflow complex. This showed me that successful ML is not about getting it right the first time, but about being disciplined enough to figure out what went wrong and continuously improve through iteration.

I used to explain the difference between supervised and unsupervised learning by simply saying one uses labelled data and the other doesn't. That was a shallow description. The real difference is much simpler and more powerful: it's about whether the model is given the "answer key" during its training.

In supervised learning, we give the model both the questions and the answers. It's like teaching a student with a study guide that has all the correct solutions. The model is constantly guided, correcting its behaviour based on the known right answer. This makes it perfect for tasks where we need a clear prediction, like forecasting a price or classifying an image.

On the other hand, unsupervised learning is like giving a student a huge pile of papers and telling them to sort and group them without any instructions. There is no answer key. The model’s job is simply to find hidden structure, patterns, or similar groups within the raw data. Since there's no pre-defined correct answer, we can't evaluate it with a simple accuracy score. This clear difference the presence or absence of the answer key helped everything fall into place.

Overfitting: Learning the Rules vs. Memorizing Mistake

The concept of overfitting became crystal clear once I saw it visually. Before, I thought a perfect score on the training data was the main goal. Now I understand that a model can achieve a perfect score by memorizing the training data, errors and all, instead of learning the general rules.

The lab's examples showed how a model can bend its prediction line to perfectly hit every single data point, even the tiny mistakes or "noise." This is a problem because the model has become too detailed and specific; it captures the noise along with the actual pattern. When we show this overfit model new, real-world data it hasn’t seen before, it struggles badly because it’s trying to apply the memorized mistakes. The model can’t generalize. This insight makes the train/test split crucial. The test data acts as a safety check, proving that the model has truly learned a general pattern that works everywhere, not just a specific set of answers for the training data.