Six-Day Intensive: What You Can Now Do

Decision Toolkit, Algorithm Map, and Next Steps

Day 6 Closing Review

The First Question: Do You Have Labels?

This single question determines your entire modelling strategy

Labels = ground truth you collected in advance. No labels: let the data self-organise.

Supervised: Regression or Classification?

Second decision: what kind of output does the task require?

Regression (continuous output)

- Predict a number on a continuous scale
- Linear Regression, Ridge, Lasso, MLP Regressor
- *Examples:* stock return tomorrow, loan default probability score

Key metrics: RMSE, R^2, MAE

Classification (discrete output)

- Predict a category or class label
- Logistic Regression, Decision Tree, Random Forest, SVM, MLP
- *Examples:* fraud / not fraud, credit approved / denied

Key metrics: Accuracy, F1, AUC-ROC

Quick test

If the output is yes/no or a named category: classification. If it is a quantity on a continuum: regression.

Most finance prediction tasks map to one of these two branches.

Day 1: Python and Data Foundations

Core skill: turning raw data into analysis-ready DataFrames

- NumPy arrays, pandas DataFrames, slicing and filtering
- Reading CSV/Excel files, handling missing values
- Exploratory analysis: `describe()`, histograms, scatter plots
- Reproducibility via `np.random.seed(42)` in every script

Finance application: Load a stock price CSV, compute daily returns, and flag anomalies (days where return $> 3\sigma$ from the mean).

Everything downstream depends on clean, well-understood data. Day 1 is the foundation.

Day 2: Statistics and Visualisation

Core skill: quantifying uncertainty and communicating patterns

- Distributions: normal, t, Poisson; mean, variance, skew, kurtosis
- Hypothesis testing: t-test, p-values, confidence intervals
- Matplotlib and Seaborn: histograms, box plots, correlation heatmaps
- Multi-panel figures for side-by-side comparison

Finance application: Test whether two trading strategies have statistically different Sharpe ratios using a two-sample t-test ($p < 0.05$).

Statistics is the language of uncertainty. Visualisation is how you communicate findings to non-technical stakeholders.

Day 3: Supervised Learning Basics

Core skill: fitting a model to labelled data and evaluating it honestly

- Train/test split, cross-validation, overfitting vs. underfitting
- Linear Regression, Ridge, Lasso regularisation
- Logistic Regression for binary classification
- Evaluation: RMSE, accuracy, confusion matrix, classification report

Finance application: Predict next-month portfolio return from factor exposures (momentum, value, size) using Ridge Regression with cross-validated α.

A model evaluated only on training data is a model that has not been tested. Always hold out a test set before you start.

Day 4: Advanced Supervised Learning and Neural Networks

Core skill: choosing complex models and understanding how they learn

- Decision Trees, Random Forest, SVM, XGBoost
- Multi-layer Perceptron: forward pass, ReLU, backpropagation, loss curves
- Regularisation: dropout, early stopping, L1/L2 weight penalties
- Scaling is mandatory before MLP training (StandardScaler)

Finance application: Train a Random Forest to classify earnings-call sentiment; use feature importances to identify the five most predictive phrases.

Neural networks learn their own feature representations. More power means more risk of overfitting.

Day 5: Unsupervised Learning and Embeddings

Core skill: finding structure without labels

- K-Means clustering: elbow curve, inertia, centroid initialisation
- PCA: explained variance ratio, component loadings, biplot
- Sentence embeddings (paraphrase-MiniLM-L3-v2): 384-dimensional vectors
- Cosine similarity for semantic ranking and retrieval

Finance application: Cluster 30 stocks by return behaviour into bonds, growth, and tech regimes; use PCA to identify the market factor (PC1).

Unsupervised methods reveal structure the labels did not tell you was there. Embeddings convert language into geometry.

Day 6: Transformers, LLMs, and RAG

Core skill: understanding and using large language models

- Word2Vec and embeddings: king $-$ man $+$ woman $=$ queen
- Attention mechanism, BERT vs. GPT, pre-training vs. fine-tuning
- RAG pipeline: query $\rightarrow$ embed $\rightarrow$ retrieve $\rightarrow$ augment $\rightarrow$ generate
- Hallucination risk and mitigation through grounding

Finance application: Build an earnings-call Q&A system by embedding 50 report chunks, retrieving the top-3 by cosine similarity, and passing them to a GPT prompt.

LLMs are pattern-completion machines trained on enormous corpora. RAG grounds them in your private, up-to-date documents.

Algorithm Cheat Sheet: Supervised Methods

Algorithm	Use when	Key param	Finance use	Red flag
Linear Reg.	Continuous target	`fit_intercept`	Predict return	Features unscaled
Ridge / Lasso	Correlated features	`alpha`	Factor model	`alpha` not tuned
Logistic Reg.	Binary class.	`C`	Default risk	Imbalanced classes
Decision Tree	Interpretability	`max_depth`	Scorecard	No depth limit
Random Forest	Tabular accuracy	`n_estimators`	Fraud detection	Slow on large n
SVM	Small data, margin	`C`, kernel	Regime class.	No feature scaling
MLP	Non-linear, large n	hidden sizes	Credit scoring	Data leakage

When in doubt: start with Logistic Regression (classification) or Ridge Regression (regression). Add complexity only when needed.

Algorithm Cheat Sheet: Unsupervised and Language Methods

Algorithm	Use when	Key param	Finance use	Red flag
K-Means	Find k groups	k, `n_init`	Stock regimes	k without elbow
Hierarchical	Unknown k	`linkage`	Sector tree	Slow on n>500
PCA	Correlated features	`n_components`	Risk factors	Not scaled first
Embeddings	Semantic similarity	Model choice	Earnings sim.	Wrong domain model
RAG	Private doc Q&A	Chunk size	Earnings Q&A	Chunk too large

PCA tip

Apply PCA before K-Means whenever you have more than 10 features. It speeds convergence and removes noise dimensions that corrupt the distance metric.

These 12 methods cover 90% of real-world quantitative finance modelling tasks.

Overfitting, Underfitting, and Class Imbalance

Overfitting

- Training accuracy high, test accuracy low
- Model memorised noise, not signal
- Fix: regularise, reduce depth, add data, early stopping, dropout

Underfitting

- Both training and test accuracy are low
- Model too simple for the task
- Fix: more features, deeper model, more iterations

Class Imbalance

- 97% of loans labelled "not fraud"
- A model that always predicts "not fraud" scores 97% accuracy
- Catches 0% of actual fraud cases
- Fix: use F1 score, AUC-ROC, or SMOTE oversampling

Signal: watch training vs. validation loss at every epoch. A widening gap is the earliest warning of overfitting.

Match your metric to the business objective. Accuracy is almost never the right metric for fraud or default prediction.

Wrong Metric and Data Leakage

Wrong Metric

- Accuracy on imbalanced data reports false confidence
- A trading strategy with positive returns but Sharpe < 0.3 is not viable in practice
- Always match the metric to what the stakeholder actually cares about

Data Leakage (the most dangerous mistake in production ML)

- Fitting `StandardScaler` on the full dataset (including test rows) leaks future statistics
- Using tomorrow's price to predict tomorrow's price is not a model: it is a tautology
- In time series: never shuffle rows before splitting into train and test
- Rule: **fit** preprocessing only on training data; **transform** both sets with those parameters

Finance example: A credit model trained on 2018 data reported 99.9% accuracy on its test set. After COVID-19 caused a sudden 30% income shock in March 2020, approvals that month defaulted at five times the predicted rate. The model had never seen a pandemic-scale income shock in its training window.

Why leakage is silent

Leakage makes models look perfect in the notebook and fail completely in production. It is the leading cause of unreproducible research results.

If your test accuracy is suspiciously high, check for leakage first.

Six Things You Can Now Do

After this intensive you can:

1. Load, clean, and explore a financial dataset with pandas and matplotlib
2. Fit and evaluate supervised models, reading the confusion matrix, F1 score, and AUC-ROC
3. Diagnose overfitting from a loss curve and apply regularisation to fix it
4. Cluster assets by return behaviour, reduce dimensions with PCA, and interpret a factor model
5. Compute cosine similarity between text documents using sentence embeddings
6. Explain the RAG architecture and prototype a document Q&A system in Python

These six skills cover the full pipeline

From raw data, through machine learning, to a deployed language model interacting with private documents in natural language.

Each skill is independently useful. Together, they form a complete quantitative finance AI toolkit.

Where to Go Next

Practice

- Reopen any Day 1–6 Colab notebook and change one hyperparameter. Observe what breaks.
- Download a real dataset from Kaggle (Finance, NLP, or Time Series track).
- Implement the credit scoring MLP on a different dataset and compare architectures.

Documentation

- scikit-learn User Guide: `scikit-learn.org/stable/user_guide.html`
- Hugging Face Quickstart: `huggingface.co/docs/transformers/quicktour`
- sentence-transformers library: `sbert.net`

Course materials (slides, notebooks, reading companions)

- `digital-ai-finance.github.io/data-science`

The best next step: apply one technique to a dataset you care about. Curiosity beats curriculum.