A Collaborative Initiative by **Open Minds Technologies** & **Discensys**

Machine Learning Course Syllabus

Comprehensive · Practical · Industry-Ready

45	40	9	5	Mon–Fri
Total Sessions	Teaching Sessions	Weeks	Sessions / Week	Schedule

Course Overview

Program Structure

This course runs across 9 weeks (45 sessions, Mon–Fri). Weeks 1–8 are 40 dedicated teaching sessions covering 16 modules in a structured, week-by-week progression. Week 9 (sessions 41–45) is the Capstone & Career Week — doubt solving, project development, a Resume & Career Readiness Workshop, final presentations, and course wrap-up.

Note: *Deep Learning is not taught in this course. The focus is mastering classical ML thoroughly — the correct foundation before tackling neural networks. Deep Learning is recommended as a dedicated follow-on track after completing this course.*

Assignment Submission — Centralised GitHub Organisation Repo

All assignments — daily, weekly, bi-weekly, and the capstone — are submitted as pull requests to the centralised course organisation repository on GitHub. Each learner works on their own named branch, pushes their work, and opens a PR for instructor review. This mirrors exactly how professional engineering teams collaborate on shared codebases.

Daily assignments are pushed to the learner's branch and a PR is opened by end of each session. Weekly assignments are submitted as a PR at the end of each week, reviewed by the instructor before the next week begins. Bi-weekly team assignments are raised from a shared team branch. The capstone project lives in its own dedicated repo under the organisation. This single centralised workflow gives instructors full visibility of every learner's progress in one place — and gives learners a real-world PR review experience from day one.

Daily Assignments (Individual)

Every session has a daily assignment — short, targeted coding tasks that reinforce the exact concept taught that day. Pushed to the learner's branch on the org repo and a PR opened by end of day. These are the primary vehicle for developing hands-on proficiency.

Weekly Assignments (Individual)

At the end of each teaching week, learners raise a PR on the org repo covering all topics from that week. The instructor reviews, leaves inline comments, and either requests changes or merges — giving learners structured, code-level feedback on their understanding.

Bi-Weekly Assignments (Team)

Every two weeks, teams push their collaborative solution to a shared team branch on the org repo and raise a PR. The team manages the branch together — dividing work, reviewing each other's commits, and resolving conflicts before the PR is submitted. Mirrors real team delivery.

Capstone Project (Team) — Week 9 (Sessions 41–45)

Teams build an end-to-end ML project from problem definition through deployment. The project lives in a dedicated repo under the organisation. Mentor reviews happen via PR comments. Certificate of completion issued upon successful delivery and final presentation.

Assessment Type	Frequency	Submission Method	Mode
Daily Assignment	Every session	Push to learner branch + PR on org repo	Individual

Assessment Type	Frequency	Submission Method	Mode
Weekly Assignment	End of each teaching week	PR on org repo — instructor reviewed	Individual
Bi-Weekly Assignment	Every 2 weeks	Team branch PR on org repo	Team
Capstone Project	Week 9 (Sessions 41–45)	Dedicated org repo + Final Presentation	Team

Weekly Module Breakdown

Week 1 | Sessions 1–5

ML Foundations, Python Toolkit & Data Preprocessing

Mon	Tue	Wed	Thu	Fri
S1	S2	S3	S4	S5
Intro to ML — types, workflow & real-world applications	NumPy & Pandas — arrays, DataFrames, data I/O & manipulation	Matplotlib, Seaborn & Scikit-Learn API orientation	Data Preprocessing I — missing values, encoding, imbalanced data	Data Preprocessing II — scaling, normalization & cleaning pipelines

Module 1: Introduction to Machine Learning

Session 1

- **What is Machine Learning and why it matters**
- **Types of ML:**
  - Supervised Learning — labelled data; model learns X → Y mapping (classification & regression)
  - Unsupervised Learning — unlabelled data; discovers patterns, clusters & structure
  - Semi-Supervised Learning — small labelled + large unlabelled dataset; reduces annotation cost
  - Self-Supervised Learning — labels derived from the data itself; foundation for representation learning
- **Note — adjacent fields (not covered in this course):**
  - Reinforcement Learning — agent learns through trial & reward; a distinct paradigm
  - Deep Learning — neural-network-based ML sub-field; computationally intensive; recommended as a dedicated follow-on course after mastering classical ML fundamentals
- **The end-to-end ML workflow:**
  - Problem definition Data collection Preprocessing
  - Model building Evaluation Deployment Monitoring
- **Real-world applications:**
  - Healthcare (diagnosis), finance (fraud), retail (recommendation)
  - Autonomous vehicles, NLP, computer vision

Module 2: Python, ML Tools & Git Workflow

Sessions 2–3

Prerequisite: Basic Python programming knowledge assumed

- **Git & GitHub — centralised org repo workflow from day one:**
  - git clone — clone the course organisation repo to your local machine
  - git checkout -b firstname-lastname/daily/session-01 — your personal branch naming convention
  - git add, git commit, git push — the daily assignment submission cycle
  - Pull Requests — open a PR from your branch to main after every assignment; instructor reviews inline
  - git pull origin main — syncing your branch before each new assignment to avoid conflicts
  - Handling merge conflicts — reading conflict markers, resolving & committing the fix

.gitignore — excluding venvs, __pycache__, .ipynb_checkpoints, .env, large data files
Why this matters: every PR is a permanent, reviewable record of your learning — visible to instructors and future employers

- **NumPy:**
  n-dimensional arrays, dtype, shape, reshape, slicing
  Vectorised operations & broadcasting — avoiding Python loops
  Linear algebra — dot product, matrix multiply, transpose
- **Pandas:**
  Series & DataFrame — creation, indexing (loc/iloc), boolean filtering
  groupby, merge/join, pivot_table, apply & lambda
  Reading CSV/Excel/JSON; handling nulls with fillna/dropna
- **Matplotlib & Seaborn:**
  Line, scatter, histogram, bar, boxplot, heatmap
  Subplots, figure sizing, colour palettes, annotation
- **Scikit-Learn:**
  Estimator API — fit(), predict(), transform(), fit_transform()
  Datasets module, train_test_split, Pipeline overview
  Understanding the consistent API pattern across all models

Module 3: Data Preprocessing
Sessions 4–5

- **Missing value strategies:**
  Mean/median/mode imputation for numeric & categorical
  KNN imputation — neighbour-based filling
  Flagging missingness as a binary feature; when to drop rows/cols
- **Encoding categorical variables:**
  Label Encoding — for ordinal categories with natural order
  One-Hot Encoding — nominal categories; avoiding dummy variable trap
  Ordinal Encoding with custom ordering
- **Imbalanced datasets:**
  SMOTE — synthetic minority oversampling technique
  Random undersampling; class_weight='balanced' in classifiers
  Evaluation implications — why accuracy alone misleads
- **Feature scaling:**
  Min-Max Normalization — scales to [0,1], sensitive to outliers
  Z-score Standardization — zero mean, unit variance
  RobustScaler — uses IQR, robust to outliers
- **Data cleaning:**
  Type conversion, duplicate detection & removal
  Outlier capping (Winsorization) at preprocessing stage
  Consistency checks, string cleaning, date parsing
- **Sklearn Pipeline:**
  Chaining transformers + estimator — prevents data leakage
  ColumnTransformer for different preprocessing per feature type

Week 2 | Sessions 6–10
Exploratory Data Analysis & Feature Engineering

Mon	Tue	Wed	Thu	Fri
S6	S7	S8	S9	S10

EDA I — descriptive stats, distributions & correlation analysis	EDA II — advanced visualization, outlier analysis & pattern discovery	Feature Engineering I — transformations & polynomial features	Feature Engineering II — interaction features, date/time & domain features	Feature Engineering III — advanced encoding, binning & EDA-to-feature workflow

Module 4: Exploratory Data Analysis (EDA) & Visualization

Sessions 6–7

- **Univariate analysis:**
  - Histograms, KDE plots — shape of distributions
  - Box plots — quartiles, IQR, whiskers, outlier dots
  - Skewness & kurtosis — asymmetry & tail weight
  - Q-Q plots — assessing normality
- **Bivariate & multivariate analysis:**
  - Scatter plots — linear vs non-linear relationships
  - Pearson & Spearman correlation — when to use each
  - Correlation heatmaps — identifying multicollinearity
  - Pairplots — simultaneous bivariate view across all features
- **Outlier detection:**
  - IQR fence method, z-score method (|z| > 3)
  - Visual detection — boxplot, scatter, 2D isolation
  - Decision framework — cap, transform, flag, or remove
- **EDA workflow:**
  - Hypothesis formulation, insight extraction
  - Communicating findings with clear, annotated visualizations

Module 5: Feature Engineering

Sessions 8–10

- **Feature transformation:**
  - Log transform — reducing right skew, stabilising variance
  - Square root & Box-Cox transforms — parametric skew correction
  - Yeo-Johnson — handles zero and negative values
- **Polynomial & interaction features:**
  - sklearn PolynomialFeatures — degree, interaction_only
  - Manual domain-motivated combinations
  - Overfitting risk with high degree — bias-variance tradeoff
- **Date/time feature extraction:**
  - Day, month, quarter, year, weekday, is_weekend
  - Cyclical encoding — sin/cos for periodic features
  - Lag features, rolling mean/std for sequential data
- **Advanced encoding:**
  - Target encoding — mean of target per category; regularisation needed
  - Frequency encoding — replace category with count/proportion
  - Binary encoding — compact for high-cardinality features
- **Binning & discretisation:**
  - Equal-width vs equal-frequency binning
  - Custom domain-based bins (age groups, income brackets)
  - pd.cut vs pd.qcut — implementation and use cases
- **Feature engineering workflow:**
  - EDA insights hypothesis new feature validate impact
  - Always validate new features with cross-validated score

Week 3 | Sessions 11–15

Feature Selection, Dimensionality Reduction & Regression

Mon	Tue	Wed	Thu	Fri
S11	**S12**	**S13**	**S14**	**S15**
Feature Selection — filter, wrapper & embedded methods	Dimensionality Reduction — PCA in depth, t-SNE & LDA	Regression I — Simple, Multiple & Polynomial Regression	Regression II — Ridge, Lasso & ElasticNet regularization	Regression III — Evaluation metrics, residual analysis & cross-validation

Module 6: Feature Selection & Dimensionality Reduction

Sessions 11–12

- **Filter methods:**
  - Pearson correlation with target — linear relevance
  - Chi-square test — categorical feature vs categorical target
  - ANOVA F-test — numeric feature vs categorical target
  - Mutual Information — captures non-linear dependencies
- **Wrapper methods:**
  - Recursive Feature Elimination (RFE) — backwards elimination with model
  - RFECV — optimal feature count via cross-validation
- **Embedded methods:**
  - Lasso (L1) — drives irrelevant coefficients to exactly zero
  - Tree feature importance — mean decrease in impurity
  - Permutation importance — model-agnostic, measures accuracy drop
- **PCA:**
  - Covariance matrix eigenvectors (directions) & eigenvalues (variance)
  - Explained variance ratio, cumulative plot, scree plot
  - Choosing n_components — 95% variance threshold
  - Limitations — linear only, loses interpretability
- **t-SNE:**
  - Non-linear reduction for 2D/3D visualisation only
  - Perplexity — balances local vs global structure
  - NOT for feature selection or downstream ML — exploration only
- **LDA:**
  - Maximises between-class scatter, minimises within-class scatter
  - Supervised — uses class labels; at most C-1 components
  - Prefer LDA over PCA when class separation is the goal

Module 7: Supervised Learning — Regression

Sessions 13–15

- **Simple & Multiple Linear Regression:**
  - OLS — minimising sum of squared residuals
  - Coefficient interpretation — unit change in X → change in Y
  - Assumptions — linearity, independence, homoscedasticity, normality of errors
  - Multicollinearity — VIF > 10 signals a problem; correlation matrix check
- **Polynomial Regression:**
  - Extending linear model with degree-n terms
  - Degree selection via validation curve — bias-variance sweet spot
- **Regularization:**
  - Ridge (L2) — shrinks all coefficients; handles multicollinearity
  - Lasso (L1) — sparse solutions; automatic feature selection
  - ElasticNet — blends L1 + L2; alpha & l1_ratio tuning
  - Choosing regularization via cross-validated grid search

- **Evaluation metrics:**
  - MAE — mean absolute error; robust to outliers
  - MSE & RMSE — penalise large errors; same units as target (RMSE)
  - R^2 — proportion of variance explained; adjusted R^2 for multiple features
  - Residual plots — checking homoscedasticity & normality
- **Cross-validation:**
  - k-Fold CV — k=5 or 10; bias-variance tradeoff in k selection
  - Stratified k-Fold — maintains class proportions
  - cross_val_score vs cross_validate — metrics vs diagnostics

Week 4 | Sessions 16–20

Classification Algorithms

Mon	Tue	Wed	Thu	Fri
S16	S17	S18	S19	S20
Classification I — Logistic Regression & decision boundaries	Classification II — Decision Trees & Random Forests	Classification III — SVM & kernel methods	Classification IV — Gradient Boosting (XGBoost, LightGBM)	Classification V — Evaluation, calibration & imbalanced classification

Module 8: Supervised Learning — Classification

Sessions 16–20

- **Logistic Regression:**
  - Sigmoid function — probability output; decision threshold
  - Binary, One-vs-Rest (OvR) & Softmax for multi-class
  - Log-loss (cross-entropy) — the loss being minimised
  - Regularization — L1/L2 via C parameter; interpretable coefficients
- **Decision Trees:**
  - Gini impurity vs entropy — splitting criteria compared
  - Depth, min_samples_split, min_samples_leaf — pruning hyperparameters
  - Feature importance from mean decrease in impurity
  - Pros: interpretable; cons: high variance, prone to overfitting
- **Random Forests:**
  - Bagging — bootstrap sampling + feature subsampling (max_features)
  - Out-of-bag (OOB) error — free validation without a holdout set
  - Aggregation — majority vote (classification), mean (regression)
  - n_estimators, max_depth, max_features — key hyperparameters
- **Support Vector Machines (SVM):**
  - Maximum-margin hyperplane — the core SVM objective
  - Soft margin — C parameter controls misclassification tolerance
  - Kernel trick — RBF, polynomial, sigmoid for non-linear boundaries
  - Support vectors — only these points define the boundary
  - SVR — regression variant; epsilon-insensitive loss
- **Gradient Boosting:**
  - Boosting concept — sequential correction of residual errors
  - XGBoost — regularized boosting; handles missing values natively
  - LightGBM — leaf-wise growth; faster on large datasets
  - Key hyperparameters: n_estimators, learning_rate, max_depth, subsample
  - Early stopping — prevent overfitting; monitor validation loss
- **Classification evaluation:**
  - Confusion matrix — TP, TN, FP, FN; visualise with heatmap
  - Precision, Recall, F1 — when each matters (fraud vs spam vs medical)
  - ROC-AUC — threshold-independent ranking quality

Precision-Recall curve — preferred for imbalanced classes
Calibration — probability reliability; Platt scaling & isotonic regression

Week 5 | Sessions 21–25

Unsupervised Learning & Anomaly Detection

Mon	Tue	Wed	Thu	Fri
S21	S22	S23	S24	S25
Clustering I — K-Means, initialization & evaluation	Clustering II — Hierarchical clustering & DBSCAN	Clustering III — Cluster validation, profiling & business interpretation	Anomaly Detection I — statistical & density-based methods	Anomaly Detection II — Isolation Forest, LOF & production monitoring

Module 9: Unsupervised Learning — Clustering

Sessions 21–23

- **K-Means Clustering:**
  - Lloyd's algorithm — assignment step & update step
  - k-Means++ initialization — smarter centroid seeding
  - Elbow method & Silhouette score — choosing optimal k
  - Sensitivity to outliers & non-spherical clusters
  - Mini-batch K-Means — scalable variant for large datasets
- **Hierarchical Clustering:**
  - Agglomerative — bottom-up; linkage criteria: single, complete, average, Ward
  - Dendrogram — visualising merge history; cutting for k clusters
  - Ward linkage — minimises within-cluster variance; most common choice
  - No need to pre-specify k; interpretable cluster tree
- **DBSCAN:**
  - Density-based — eps (neighbourhood radius) & min_samples
  - Core points, border points, noise points — the three categories
  - Finds arbitrary-shaped clusters; robust to outliers
  - Limitations: struggles with varying density clusters
- **Cluster validation:**
  - Silhouette coefficient — intra vs inter-cluster distance
  - Davies-Bouldin index — lower is better
  - Calinski-Harabasz score — ratio of between to within-cluster dispersion
  - Visual validation — PCA/t-SNE 2D projection of clusters
- **Cluster profiling & business interpretation:**
  - Per-cluster feature statistics — mean, distribution, dominant values
  - Naming clusters based on dominant characteristics
  - Customer segmentation, product grouping & anomaly buckets

Module 10: Anomaly Detection

Sessions 24–25

- **Statistical methods:**
  - Z-score thresholding — assumes normality
  - IQR fences — robust; no distributional assumption
  - Grubbs test, GESD — formal outlier tests
- **Density-based methods:**
  - Local Outlier Factor (LOF) — local density comparison
  - n_neighbors — key hyperparameter; neighbourhood size
- **Isolation Forest:**
  - Random partitioning — anomalies require fewer splits to isolate

contamination parameter — expected anomaly fraction
Efficient on high-dimensional data

- **Multivariate anomaly detection:**
  Mahalanobis distance — accounts for feature correlations
  One-Class SVM — learns a tight boundary around normal data
- **Monitoring in production:**
  Rolling z-score — real-time flagging of metric drift
  Statistical process control charts (Shewhart charts)
  Alerting thresholds — balancing false positives vs misses

Week 6 | Sessions 26–30

Time Series & Model Optimization

Mon	Tue	Wed	Thu	Fri
S26	S27	S28	S29	S30
Time Series I — stationarity, decomposition & smoothing	Time Series II — ARIMA, SARIMA & forecasting	Time Series III — ML-based forecasting & feature engineering for TS	Model Optimization — hyperparameter tuning, cross-validation & pipelines	Model Explainability — SHAP, feature importance & bias-variance analysis

Module 11: Time Series Analysis & Forecasting

Sessions 26–28

- **Time series fundamentals:**
  Components — trend, seasonality, cyclicality, noise
  Additive vs multiplicative decomposition
  Autocorrelation (ACF) & partial autocorrelation (PACF) plots
  Stationarity — ADF test; differencing to remove trend & seasonality
- **Classical forecasting models:**
  SMA & EMA — simple & exponential moving averages
  Holt-Winters (Triple Exponential Smoothing) — trend & seasonality
  ARIMA (p,d,q) — AR terms, differencing, MA terms; AIC/BIC model selection
  SARIMA — seasonal extension; (P,D,Q,m) seasonal component
  auto_arima from pmdarima — automatic order selection
- **ML-based time series forecasting:**
  Feature engineering for TS — lag features, rolling statistics, time-based features
  Train/test split — always time-ordered; never shuffle TS data
  TimeSeriesSplit — walk-forward cross-validation
  Random Forest & XGBoost on tabular TS features
  Forecast horizon — one-step vs multi-step prediction strategies
- **Evaluation:**
  MAE, RMSE, MAPE — and when MAPE fails (near-zero actuals)
  Residual diagnostics — Ljung-Box test for autocorrelation in residuals

Module 12: Model Optimization & Explainability

Sessions 29–30

- **Hyperparameter tuning:**
  Grid Search — exhaustive; guarantees optimal in grid
  Random Search — samples efficiently; better with large spaces
  Bayesian Optimization (Optuna) — builds surrogate model; smarter search
  Halving Search — resource-efficient early elimination
- **Cross-validation strategies:**
  Stratified k-Fold for classification — preserves class balance

Group k-Fold — when observations are grouped (patient, user)
Repeated k-Fold — reduces evaluation variance
Nested CV — unbiased hyperparameter selection + evaluation

- **Pipeline best practices:**
  Full pipeline — preprocessor + model; prevents leakage
  Pipeline.set_params() — tuning nested components
  Persisting pipelines with joblib — production-ready serialization
- **Bias-Variance Analysis:**
  Learning curves — diagnosing underfitting vs overfitting
  Validation curves — effect of single hyperparameter on train/val score
  Practical remedies — regularization, more data, simpler model
- **Model Explainability with SHAP:**
  SHapley Additive exPlanations — game-theory-based feature attribution
  SHAP values — each feature's marginal contribution to a prediction
  Global explanations — mean |SHAP| bar plot; beeswarm plot
  Local explanations — waterfall plot; explaining a single prediction
  TreeExplainer — fast exact SHAP for tree-based models
  KernelExplainer — model-agnostic SHAP; slower, any model
  Industry use — regulatory compliance, debugging, trust-building

Week 7 | Sessions 31–35

Classical ML Completion & NLP Fundamentals with ML

Mon	Tue	Wed	Thu	Fri
S31	S32	S33	S34	S35
KNN & Naive Bayes — algorithms, distance metrics, probabilistic models	Advanced Ensembles — Stacking, Blending & Gaussian Mixture Models (GMM)	NLP I — text preprocessing, tokenization, stemming & lemmatization	NLP II — BOW, TF-IDF, n-grams & text classification with ML	NLP III — Sentiment analysis, end-to-end text ML pipeline & spaCy/NLTK

Module 13: Classical ML Algorithms & Advanced Ensembles

Sessions 31–32

- **K-Nearest Neighbours (KNN):**
  Instance-based learning — no explicit training phase
  Distance metrics — Euclidean, Manhattan, Minkowski; when each applies
  Choosing k — odd values for binary; cross-validated selection
  Curse of dimensionality — why KNN degrades in high dimensions
  Weighted KNN — distance-inverse weighting for smoother boundaries
  KNN for regression — predict mean of k nearest neighbours
- **Naive Bayes:**
  Bayes' theorem — posterior $\propto$ likelihood $\times$ prior
  Naive assumption — feature independence given the class
  Gaussian NB — continuous features; assumes normal distribution per class
  Multinomial NB — count data; natural fit for text (word counts)
  Bernoulli NB — binary features; document-level presence/absence
  Laplace (additive) smoothing — handles zero-count categories
  Strong baseline for text; fast & interpretable
- **Advanced Ensemble Methods:**
  Stacking — train Level-1 models; meta-learner on their OOF predictions
  Blending — holdout-based stacking; simpler but slightly less data-efficient
  Voting Classifier — hard vote (majority) vs soft vote (probability average)
  When stacking helps — diverse base models with complementary errors

- **Gaussian Mixture Models (GMM):**
  - Probabilistic clustering — soft assignments (responsibilities)
  - EM algorithm — Expectation step (assign) & Maximisation step (update)
  - Covariance types — full, tied, diagonal, spherical
  - Model selection — BIC/AIC for choosing n_components
  - Generates probability of cluster membership; richer than K-Means

Module 14: NLP Fundamentals with Machine Learning

Sessions 33–35

- **Text preprocessing pipeline:**
  - Lowercasing, punctuation removal, whitespace normalisation
  - Tokenization — word-level & sentence-level
  - Stopword removal — NLTK stopword list; domain-aware exceptions
  - Stemming (Porter/Snowball) — rule-based suffix stripping
  - Lemmatization (spaCy) — context-aware; returns dictionary form
  - Regex patterns — extracting mentions, hashtags, URLs, dates
- **Text representation:**
  - Bag of Words (BOW) — sparse term-count matrix
  - TF-IDF — term frequency × inverse document frequency; reduces common word dominance
  - N-grams — unigrams, bigrams, trigrams; capturing local context
  - Vocabulary size control — max_features, min_df, max_df
  - Sparse matrix handling — scipy.sparse; memory-efficient storage
- **Text classification with ML:**
  - Multinomial Naive Bayes — strong text baseline; fast & interpretable
  - Logistic Regression on TF-IDF — regularized; highly competitive
  - LinearSVC — efficient SVM for text; handles high-dimensional sparse input
  - Evaluation — accuracy, F1 (macro/micro), classification report
- **Sentiment analysis:**
  - Rule-based baseline — VADER (Valence Aware Dictionary for Sentiment Reasoning)
  - ML-based — TF-IDF + Logistic Regression / Naive Bayes on labelled reviews
  - Dataset sources — IMDB, Amazon reviews, Twitter sentiment
  - Handling negation — 'not good' vs 'good'; n-gram capture
- **Word embeddings (conceptual):**
  - Word2Vec — CBOW & Skip-gram; semantically similar words cluster together
  - GloVe — global co-occurrence statistics
  - Pre-trained embeddings as features for downstream ML classifiers
  - Why embeddings > BOW — captures semantic similarity, not just frequency
- **End-to-end NLP ML pipeline:**
  - Text → preprocess → vectorize → ML model → evaluate
  - Pipeline integration — sklearn Pipeline with TfidfVectorizer + classifier
  - spaCy & NLTK — practical usage for preprocessing at scale
  - Deployment consideration — save vectorizer + model together

Week 8 | Sessions 36–40

Model Deployment & Capstone Kickoff

Mon	Tue	Wed	Thu	Fri
S36	S37	S38	S39	S40
Deployment I — model serialization, FastAPI setup & REST API design	Deployment II — API testing, error handling & authentication	Deployment III — Docker containerisation & cloud deployment	Deployment IV — monitoring, logging & model drift detection	Capstone Kickoff — project brief, team formation & dataset scoping

Module 15: Model Deployment & MLOps Essentials

Sessions 36–39

- **Model serialization:**
  - joblib — recommended for sklearn models; handles large numpy arrays
  - pickle — standard Python; version-sensitive
  - Versioning saved models — naming convention with date/version tag
  - Saving the full pipeline — preprocessor + model together
- **FastAPI for ML serving:**
  - Project structure — main.py, schemas, model loading on startup
  - Pydantic schemas — request/response validation
  - POST /predict endpoint — input validation, inference, structured response
  - Async endpoints — non-blocking inference for concurrent requests
  - Swagger UI at /docs — auto-generated interactive API documentation
- **API robustness:**
  - Input validation errors — 422 Unprocessable Entity; informative messages
  - Exception handling — global exception handler; graceful degradation
  - API key authentication — header-based simple auth
  - Testing with pytest + httpx — automated endpoint tests
- **Docker & containerisation:**
  - Dockerfile — base image, COPY, RUN, EXPOSE, CMD
  - docker build & docker run — building and running the ML container
  - Port mapping — container port to host port
  - .dockerignore — excluding model artefacts & virtual envs if not needed
  - Multi-stage builds — smaller production images
- **Cloud deployment:**
  - Render / Railway — simple PaaS; push-to-deploy from GitHub
  - AWS EC2 basics — SSH, security groups, running FastAPI with uvicorn
  - Environment variables — never hardcode secrets; use .env + python-dotenv
  - Health check endpoint — /health for load balancer & uptime monitoring
- **Monitoring & MLOps basics:**
  - Logging — structured JSON logs; request/response audit trail
  - Data drift — feature distribution shift over time; KS test
  - Concept drift — model performance degradation; retraining triggers
  - Model versioning — MLflow experiment tracking overview
  - CI/CD for ML — automated test & deploy on push

Module 16: Capstone Project Kickoff

Session 40

- **Capstone project overview:**
  - Problem selection — real-world dataset; business framing required
  - Team roles — project lead, EDA, modelling, deployment, documentation
  - Deliverables — Jupyter notebook, FastAPI app, org GitHub repo, 10-min presentation
  - Evaluation rubric — data quality, model performance, code quality, presentation
- **Scoping & repo setup:**
  - Dataset selection & feasibility check
  - Problem definition — classification, regression, clustering or forecasting
  - Define success metric aligned to business goal
  - Capstone repo created under the organisation — team members added as collaborators
  - Branching strategy agreed — one branch per team member; PRs for integration

Week 9 | Sessions 41–45

Capstone & Career Week

Mon	Tue	Wed	Thu	Fri
S41	S42	S43	S44	S45
Capstone Work — mentored project development & doubt-solving (session 1)	Capstone Work — mentored project development & doubt-solving (session 2)	Resume, Portfolio & Career Readiness Workshop	Final Presentations — team demos, Q&A & peer review	Course Wrap-up — feedback, certificates & next steps

Capstone Project (Sessions 41–45)

- **End-to-end ML project delivery:**
  - Full pipeline — data ingestion, preprocessing, feature engineering, modelling, deployment
  - All work on the organisation GitHub repo — PRs used for mentor review & integration
  - Mentor-led doubt-solving sessions (S41 & S42) — feedback delivered via PR comments
  - Mid-capstone checkpoint — progress review; mentor merges or requests changes
- **Final Presentation (Session 44):**
  - 10-minute team demo + 5-minute Q&A
  - Present: problem statement, data, approach, results, limitations, deployment
  - Peer review — structured feedback form for cross-team assessment
  - Certificate of Completion issued upon successful delivery

Resume, Portfolio & Career Readiness Workshop (Session 43)

- **ML Resume Structure:**
  - One-page format for 0–3 years experience; two-page maximum for senior roles
  - Skills section — programming languages, ML libraries, tools (Git, Docker, cloud)
  - Projects section — problem → approach → result → impact (quantified)
  - ATS optimisation — keyword alignment with job descriptions
- **GitHub Portfolio:**
  - By this point: 9 weeks of daily PRs on the org repo — a live, reviewable track record
  - Capstone repo under the org — link it prominently; shows real team collaboration
  - Personal profile README — pinned repos, skills summary, contact links
  - Contribution graph — consistent activity signals reliability to recruiters
  - README structure for each project — problem, dataset, approach, results, how to run
- **LinkedIn Optimisation:**
  - Headline — 'ML Engineer | Python | Scikit-Learn | FastAPI' style
  - About section — 3–5 sentences: background, skills, what you're building
  - Featured section — link to the org GitHub, a deployed project, or blog post
  - Skills & endorsements — request endorsements from cohort peers
- **Interview Guidance:**
  - Technical rounds — expect to explain algorithms, tradeoffs & hyperparameters
  - ML case studies — problem framing, metric selection, pipeline design
  - Coding rounds — Python, pandas, sklearn; LeetCode medium-level data manipulation
  - Behavioural — STAR method; align stories to ML project experience
  - Common questions: bias-variance, overfitting fixes, class imbalance, model selection