OPENMINDS
ONE STOP SOLUTIONS FOR YOUR CAREER

x DISCENSYS

Openminds Technologies

Data Science Course

- Foundations of Data Science
- Python for Data Science
- Statistics for Data Science
- Data Visualization
- Machine Learning for Data Science
- Data Wrangling & Engineering
- Big Data Tools
- Time Series Analysis
- Advanced Topics
- Deep Learning for Data Science
- MLOps & Deployment
- Business & Communication
- Projects & Case Studies
- Industry Case Studies & Portfolio Building

openmindstechnologies.com

Plot No:604/A, Nilgiri Block, Aditya Enclave, 6th Floor,
Ameerpet, Besides Metro Station

+91 95421 73327, 98496 83033

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

Data Science 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 13 modules -- from the foundations of data science and Python through statistics, visualisation, machine learning, data wrangling, big data tools, time series analysis, advanced topics, deep learning, MLOps, business communication, and real-world projects. Week 9 (sessions 41-45) is the Capstone & Career Week -- team project delivery, a Resume & Career Readiness Workshop, final presentations, and course wrap-up.

Prerequisite: *Basic Python programming is required -- variables, loops, functions, and basic file I/O. No prior statistics, mathematics, or data science knowledge is needed; this course builds from first principles.*

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 data science teams collaborate on shared analysis codebases.

Daily assignments are pushed to the learner's branch and a PR is opened by end of each session. Weekly assignments cover all topics from that week and are reviewed by the instructor with inline code comments. Bi-weekly team assignments are raised from a shared team branch. The capstone project lives in its own dedicated repo under the organisation.

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 building hands-on proficiency with Python, pandas, scikit-learn, and visualisation libraries.

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 data science implementations.

Bi-Weekly Assignments (Team)

Every two weeks, teams push their collaborative data science 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 analysis notebooks, and resolving conflicts. Mirrors real team data science delivery.

Capstone Project (Team) -- Week 9 (Sessions 41-45)

Teams build a complete, end-to-end data science project -- from business problem definition through EDA, modelling, and a deployed Streamlit dashboard with business insights and actionable recommendations. The project lives in a dedicated repo under the organisation. 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

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

Foundations of Data Science & Python

Mon	Tue	Wed	Thu	Fri
S1	**S2**	**S3**	**S4**	**S5**
Foundations I -- What is Data Science; the DS lifecycle; types of data	Foundations II -- data collection methods; data ethics & privacy; GDPR basics	Python I -- NumPy; arrays, vectorised operations, broadcasting, indexing	Python II -- Pandas; DataFrames, Series, merging, groupby, reshaping	Python III -- data cleaning; EDA with pandas; first end-to-end analysis notebook

Module 1: Foundations of Data Science

Sessions 1-2

- **What is Data Science:**
  - Data science extracts insights and value from data using statistics, programming, and domain knowledge
  - The DS workflow: define problem -> collect data -> clean -> analyse -> model -> communicate -> deploy
  - DS vs data analytics vs ML engineering vs data engineering -- different roles, different skill sets
  - Where DS creates value: product personalisation, demand forecasting, fraud detection, medical diagnostics
  - The data science team: data scientist, data engineer, ML engineer, analyst -- how they collaborate
- **Data Science Lifecycle:**
  - Business understanding -- define the problem, success metrics, and stakeholder needs before touching any data
  - Data acquisition -- identify data sources, assess quality and availability, understand governance constraints
  - Data preparation -- cleaning, transformation, feature engineering; typically 60-80% of a data scientist's time
  - Modelling -- select algorithms, train, tune hyperparameters, validate on held-out data
  - Deployment and monitoring -- move model to production, track performance, retrain when performance drifts
- **Types of Data:**
  - Structured data -- rows and columns (CSV, SQL tables); easy to query and model directly
  - Semi-structured data -- some structure but not tabular (JSON, XML, HTML); requires parsing
  - Unstructured data -- no predefined format (text, images, audio, video); needs feature extraction or deep learning
  - Categorical data -- nominal (no order: city, colour) vs ordinal (ordered: rating, education level)
  - Quantitative data -- continuous (any value in a range: temperature, price) vs discrete (countable: transactions)
- **Data Collection Methods:**
  - Primary data -- surveys, experiments, sensors, web scraping; collected directly for the purpose at hand
  - Secondary data -- existing datasets (Kaggle, UCI, government portals, APIs); check for bias and staleness
  - Web scraping -- BeautifulSoup, Scrapy; extract structured data from HTML; respect robots.txt and terms of service
  - APIs -- structured, authorised access to data (REST APIs: OpenWeatherMap, financial data, social media)
  - Data quality dimensions: completeness, consistency, accuracy, timeliness, uniqueness; assess before modelling
- **Data Ethics and Privacy:**
  - Informed consent -- data subjects should know how their data is collected, stored, and used
  - GDPR (EU) and CCPA (California) -- legal frameworks; right to access, deletion, and data portability
  - Bias in data -- historical bias, sampling bias, measurement bias; biased data produces biased models
  - Fairness metrics -- demographic parity, equal opportunity; choice depends on use case and stakes
  - Data anonymisation -- remove or mask PII (Personally Identifiable Information); k-anonymity concepts

Module 2: Python for Data Science

Sessions 3-5

- **NumPy:**
  - NumPy is the foundation of scientific computing in Python -- provides fast N-dimensional array (ndarray) objects
  - Array creation -- np.array(), np.zeros(), np.ones(), np.arange(), np.linspace(), np.random.randn()
  - Vectorised operations -- element-wise arithmetic without loops; 100x faster than pure Python for numerical data
  - Broadcasting -- arithmetic between arrays of different shapes; NumPy expands dimensions automatically to match
  - Indexing and slicing -- arr[2:5], arr[arr > 0] boolean masks; fancy indexing; np.where() for conditional selection
  - Common operations -- np.sum(), np.mean(), np.std(), np.reshape(), np.concatenate(), np.dot() for matrix math
- **Pandas:**
  - Pandas provides DataFrame (2D table) and Series (1D) built on NumPy; the core tool for data analysis in Python
  - Reading data -- pd.read_csv(), pd.read_excel(), pd.read_json(), pd.read_sql(); specify dtypes and parse_dates
  - Indexing -- .loc[] (label-based), .iloc[] (integer-based), boolean filtering; set_index() for custom index
  - Merging -- pd.merge() with inner, left, right, outer join; pd.concat() for stacking; join on single or multiple keys
  - GroupBy -- group by one or more columns; apply aggregate functions (sum, mean, count, custom agg); split-apply-combine
- **Data Manipulation:**
  - String operations -- .str.lower(), .str.replace(), .str.contains(), .str.split(); clean messy text columns
  - DateTime handling -- pd.to_datetime(), .dt.year, .dt.month, .dt.dayofweek; resample() for time-based aggregation
  - Apply and map -- .apply() for row/column functions, .map() for element-wise value replacement
  - Window functions -- .rolling(7).mean() for moving average; .shift() for lag features; .pct_change() for growth rates
  - Reshaping -- pivot_table(), melt(), stack(), unstack(); convert between wide and long format for analysis
- **Data Cleaning:**
  - Identifying issues -- .info(), .describe(), .isnull().sum(), .duplicated().sum(), .value_counts() for each column
  - Handling missing values -- .dropna() when very few missing; .fillna(mean/median/mode); forward fill for time series
  - Handling duplicates -- .drop_duplicates(); check for near-duplicates in text columns using string similarity
  - Outlier detection -- IQR method (Q1 - 1.5*IQR, Q3 + 1.5*IQR); Z-score (|z| > 3); visualise with boxplots first
  - Data type correction -- .astype(); pd.to_numeric(errors='coerce') converts non-numeric strings to NaN safely
- **Exploratory Data Analysis (EDA):**
  - EDA goal -- understand distribution, spot anomalies, discover relationships before any modelling begins
  - Univariate analysis -- histograms, boxplots, value_counts() for each feature; check skewness and outliers
  - Bivariate analysis -- scatter plots, correlation heatmap (df.corr()); cross-tabulation for categorical pairs
  - Correlation -- Pearson (linear), Spearman (rank-based, works for non-linear); heatmap with seaborn
  - 5 EDA questions: What is the shape? What are the distributions? What is missing? What correlates? What surprises?

Week 2 | Sessions 6-10

Statistics for Data Science & Visualisation -- Part 1

Mon	Tue	Wed	Thu	Fri
S6	**S7**	**S8**	**S9**	**S10**
Statistics I -- descriptive stats; central tendency; spread; skewness & kurtosis	Statistics II -- probability distributions; normal, binomial, Poisson; CLT	Statistics III -- inferential stats; sampling; z-test; t-test; hypothesis testing; p-values	Statistics IV -- confidence intervals; effect size; statistical power; multiple testing	Visualisation I -- Matplotlib; figure anatomy; line, bar, scatter, histogram; customisation

Module 3: Statistics for Data Science

Sessions 6-9

- **Descriptive Statistics:**
  - Measures of central tendency: mean (sensitive to outliers), median (robust), mode (most frequent value)
  - Measures of spread: range, variance, standard deviation, IQR (Q3-Q1); std = average deviation from mean
  - Shape of distribution: skewness (asymmetry -- positive skew has long right tail), kurtosis (tail heaviness)

Five-number summary: min, Q1, median, Q3, max; forms the basis of boxplots; reveals spread and outliers
Frequency tables and crosstabs -- for categorical variables; pd.crosstab() for two-way summaries

- **Probability Distributions:**

Normal (Gaussian) -- symmetric bell curve; defined by mean and std; empirical rule: 68-95-99.7%
Binomial -- models number of successes in n independent trials with probability p; use for conversion rates
Poisson -- models count of rare events in a fixed interval (calls per hour, defects per unit); parameter lambda
Uniform -- all outcomes equally likely; used in simulation and random sampling
Central Limit Theorem -- sample means follow normal distribution as n grows, regardless of population shape; foundation of inferential statistics

- **Inferential Statistics & Hypothesis Testing:**

Population vs sample -- we observe a sample; inference lets us draw conclusions about the full population
Null hypothesis (H0) vs alternative hypothesis (H1); hypothesis test decides whether to reject H0
z-test -- population std known; z = (x_bar - mu) / (sigma / sqrt(n)); compare to z-table critical value
t-test -- population std unknown (the common case); use sample std; t-distribution with n-1 degrees of freedom
p-value -- probability of observing data this extreme if H0 is true; reject H0 if p < alpha (usually 0.05)
Type I error (false positive): rejecting H0 when true; probability = alpha; Type II error: failing to reject false H0

- **Confidence Intervals & Statistical Power:**

Confidence interval -- CI = x_bar +/- z * (sigma / sqrt(n)); range that likely contains the true parameter
Interpretation -- a 95% CI means 95 out of 100 such intervals will contain the true population mean
Effect size -- Cohen's d; measures practical significance; a result can be statistically significant but trivially small
Statistical power -- probability of correctly detecting an effect when one exists; aim for power >= 0.80
Multiple testing -- running 20 tests at alpha=0.05 expects 1 false positive; apply Bonferroni correction or FDR

Module 4: Data Visualisation

Sessions 10-12

- **Matplotlib:**

Matplotlib is the foundational Python plotting library; all other visualisation libraries build on it
Figure and Axes -- fig, ax = plt.subplots(); figure is the canvas, axes is the plot area; multiple subplots with gridspec
Core plot types -- ax.plot() (line), ax.bar() (bar chart), ax.scatter() (scatter), ax.hist() (histogram)
Customisation -- labels (ax.set_xlabel, ax.set_title), tick marks, colour, linestyle, marker, legend placement
Saving -- plt.savefig("plot.png", dpi=300, bbox_inches="tight"); save before plt.show() in scripts

Week 3 | Sessions 11-15

Visualisation (completion) & Machine Learning -- Part 1

Mon	Tue	Wed	Thu	Fri
S11	**S12**	**S13**	**S14**	**S15**
Visualisation II -- Seaborn; distribution, categorical & relational plots; heatmaps; pairplots	Visualisation III -- Plotly interactive charts; data storytelling; dashboard basics with Streamlit	ML I -- supervised learning: regression (linear, polynomial, Ridge, Lasso); evaluation metrics	ML II -- supervised learning: classification (logistic regression, decision trees, random forests, SVM)	ML III -- unsupervised learning: k-means, hierarchical clustering, DBSCAN, PCA

Module 4: Data Visualisation -- continued

Sessions 11-12

- **Seaborn:**

Seaborn builds on Matplotlib with statistical plots and cleaner default aesthetics; integrates with Pandas DataFrames
Distribution plots -- sns.histplot(), sns.kdeplot(), sns.boxplot(), sns.violinplot(); reveal shape and spread
Categorical plots -- sns.barplot(), sns.countplot(), sns.stripplot(); compare values across groups
Relational plots -- sns.scatterplot(), sns.lineplot(); sns.regplot() adds a regression line automatically
Heatmap -- sns.heatmap(df.corr(), annot=True, cmap="coolwarm"); essential for visualising feature correlations
Pairplot -- sns.pairplot(df, hue="target"); scatter matrix of all feature pairs; fast EDA for structured data

- **Plotly & Storytelling with Data:**

Plotly -- interactive Python charts; hover tooltips, zoom, filter; export to HTML for sharing without code
Plotly Express -- px.scatter(), px.bar(), px.line(), px.choropleth(); one-line interactive charts from DataFrames
Streamlit -- st.plotly_chart(), st.slider(), st.selectbox(); deploy interactive DS apps in minutes with pure Python
Storytelling: context (who is the audience, what decision is being made), message (one clear insight per chart)
Chart selection: comparison -> bar; trend -> line; part-of-whole -> stacked bar; relationship -> scatter; distribution -> histogram

Module 5: Machine Learning for Data Science

Sessions 13-17

- **Supervised Learning -- Regression:**
  - Regression predicts a continuous numeric output (price, temperature, sales) from input features
  - Linear regression -- fit a line minimising sum of squared residuals (OLS); assumptions: linearity, homoscedasticity
  - Polynomial regression -- add polynomial features (x^2, x^3) to model non-linear relationships; risk of overfitting
  - Ridge (L2) -- adds lambda * sum(w^2) penalty; shrinks all coefficients; reduces overfitting; always keep all features
  - Lasso (L1) -- adds lambda * sum(|w|) penalty; drives some coefficients to exactly zero; performs feature selection
  - Evaluation metrics: MAE, MSE, RMSE, R-squared (proportion of variance explained); choose based on business context
- **Supervised Learning -- Classification:**
  - Classification predicts a discrete class label (spam/not spam, churn/no churn, disease/healthy)
  - Logistic regression -- models P(y=1|X) = sigmoid(wX + b); decision boundary is linear; output is a probability
  - Decision trees -- split features to maximise information gain or reduce Gini impurity; interpretable; prone to overfit
  - Random forests -- ensemble of trees on bootstrap samples with random feature subsets; reduces variance significantly
  - Gradient boosting -- XGBoost, LightGBM; trees built sequentially; each corrects the previous; best for tabular data
  - SVM -- find the margin-maximising hyperplane between classes; kernel trick enables non-linear boundaries
- **Unsupervised Learning:**
  - k-Means -- assign n points to k clusters; iteratively update centroids; choose k with elbow method or silhouette score
  - Hierarchical clustering -- build a dendrogram by merging nearest clusters; no need to specify k upfront
  - DBSCAN -- density-based; finds clusters of arbitrary shape; identifies noise points; no k required
  - PCA (Principal Component Analysis) -- reduce dimensions by projecting onto directions of maximum variance
  - Explained variance ratio -- select components that collectively explain 90%+ of variance; scree plot for visualisation
- **Model Evaluation:**
  - Train / validation / test split -- tune hyperparameters on validation set; report final metrics on test set only
  - k-fold cross-validation -- train k times; each fold serves as validation once; reliable generalisation estimate
  - Classification metrics -- accuracy (misleading for imbalanced), precision, recall, F1-score, ROC-AUC
  - Confusion matrix -- TP, TN, FP, FN; precision = TP/(TP+FP); recall = TP/(TP+FN); choose threshold based on cost
  - Overfitting vs underfitting -- high train / low val accuracy = overfit (regularise); low train = underfit (more capacity)

Week 4 | Sessions 16-20

Machine Learning (completion) & Data Wrangling

Mon	Tue	Wed	Thu	Fri
S16	**S17**	**S18**	**S19**	**S20**
ML IV -- model evaluation continued; cross-validation; ROC-AUC; handling class imbalance	ML V -- feature engineering & model selection; encoding, scaling, pipelines, GridSearchCV	Data Wrangling I -- handling missing data; MCAR/MAR/MNAR; imputation strategies	Data Wrangling II -- data transformation; encoding; ETL pipelines; Airflow intro	Data Wrangling III -- SQL for DS; window functions; large datasets; Parquet & chunking

Module 5: Machine Learning -- continued

Sessions 16-17

- **Handling Class Imbalance:**
  - Class imbalance -- when one class is rare (fraud: 0.1%); accuracy is misleading; focus on precision and recall
  - SMOTE (Synthetic Minority Oversampling Technique) -- generate synthetic minority samples; improves recall
  - Undersampling -- randomly remove majority class samples; fast; loses data; works well when majority is very large
  - Class weights -- pass class_weight="balanced" to sklearn classifiers; penalises errors on minority class more
  - Threshold tuning -- adjust the decision threshold (default 0.5); plot precision-recall curve to find the right threshold
- **Feature Engineering & Model Selection:**
  - Feature encoding -- one-hot for nominal; ordinal encoding for ordered; target encoding for high-cardinality columns
  - Feature scaling -- StandardScaler (z-score), MinMaxScaler (0 to 1); required for SVM, KNN, logistic regression
  - Creating new features -- ratios, differences, log transforms, polynomial interactions; domain knowledge drives this
  - sklearn Pipeline -- chain preprocessing and model into one object; prevents data leakage during cross-validation
  - Hyperparameter tuning -- GridSearchCV (exhaustive), RandomizedSearchCV (faster); Optuna for Bayesian optimisation

Module 6: Data Wrangling & Engineering

Sessions 18-20

- **Handling Missing Data:**
  - Understand why data is missing: MCAR (missing completely at random), MAR (missing at random), MNAR (not at random)
  - If MCAR: safe to drop rows; if MAR: imputation is valid; if MNAR: need to model the missingness mechanism itself
  - Mean/median imputation -- fast; does not introduce information; distorts correlations between features
  - KNN imputation -- use k nearest neighbours to estimate missing values; preserves feature correlations better
  - IterativeImputer (MICE) -- model each feature as a function of the others; gold standard for important variables
- **Data Transformation:**
  - Log transform -- for right-skewed data (income, population); compresses large values; reduces skewness
  - Box-Cox transform -- generalisation of log; automatically finds the best transformation parameter lambda
  - Binning -- convert continuous to categorical (age groups, income brackets); useful for non-linear relationships
  - Date feature extraction -- year, month, day, day-of-week, hour, is_weekend; powerful for retail and time series data
  - Scaling vs normalisation -- StandardScaler for Gaussian assumptions; MinMaxScaler for bounded-range requirements
- **ETL Pipelines:**
  - ETL = Extract, Transform, Load; the backbone of data engineering; moves data from sources to analytical stores
  - Extract -- pull data from databases (SQL), APIs, files (CSV/JSON/Parquet), streaming sources (Kafka)
  - Transform -- clean, normalise, join, aggregate; apply business rules; produce a consistent analytical layer
  - Load -- write to data warehouse (Snowflake, BigQuery, Redshift) or data lake; choose format (Parquet for analytics)
  - Apache Airflow -- define DAGs (Directed Acyclic Graphs) of ETL tasks; schedule, monitor, and retry on failure
- **SQL & Working with Large Datasets:**
  - SQL fundamentals -- SELECT, WHERE, GROUP BY, HAVING, JOIN (inner, left, right, full outer), subqueries
  - Window functions -- ROW_NUMBER(), RANK(), LAG(), LEAD(), SUM() OVER (PARTITION BY ...); essential for DS analysis
  - SQL in Python -- pd.read_sql() with SQLAlchemy connection; pull query results directly into a DataFrame
  - Chunking large files -- pd.read_csv(chunksize=10000); process without loading everything into memory at once
  - Parquet format -- columnar storage; 10x smaller than CSV; much faster for analytics queries; use pyarrow or fastparquet

Week 5 | Sessions 21-25

Big Data Tools & Time Series -- Part 1

Mon	Tue	Wed	Thu	Fri
S21	S22	S23	S24	S25

Big Data I -- intro to Big Data; Hadoop ecosystem; HDFS; MapReduce concepts	Big Data II -- Apache Spark; PySpark DataFrames; transformations; Spark MLlib	Big Data III -- distributed computing; data lakes vs warehouses; Delta Lake; cloud DS platforms	Time Series I -- trend, seasonality, decomposition; ACF/PACF; stationarity; ADF test	Time Series II -- ARIMA models; model identification; SARIMA for seasonal data

Module 7: Big Data Tools

Sessions 21-23

- **Introduction to Big Data & Hadoop:**
  - The 3Vs of Big Data: Volume (too large for one machine), Velocity (streaming), Variety (structured + unstructured)
  - Hadoop ecosystem -- HDFS (distributed file system), MapReduce (distributed processing), YARN (resource manager)
  - HDFS -- files split into 128MB blocks; replicated 3x for fault tolerance; write-once, read-many access pattern
  - MapReduce -- Map: apply function to each record in parallel; Reduce: aggregate results; slow but fault-tolerant
  - When to use: Hadoop for TB+ scale batch jobs; Pandas for GB scale on a single machine; Spark for in-memory speed
- **Apache Spark:**
  - Spark processes data in-memory; up to 100x faster than MapReduce; unified engine for batch, streaming, and ML
  - RDDs (Resilient Distributed Datasets) -- low-level; partitioned across nodes; lazy transformations, eager actions
  - Spark DataFrames -- higher-level API; similar to Pandas but distributed; Catalyst optimizer for query planning
  - PySpark -- SparkSession, df.filter(), df.groupBy(), df.agg(), df.write.parquet(); Python API for Spark
  - Spark MLlib -- distributed ML: linear regression, random forests, k-means, pipeline API; scales to billions of rows
- **Distributed Computing, Data Lakes & Cloud Platforms:**
  - Distributed computing -- split data and computation across multiple nodes; horizontal scaling; fault tolerance via replication
  - Data Lake -- store raw data in original format at scale (S3, GCS, Azure Data Lake); schema-on-read approach
  - Data Warehouse -- structured, processed data for analytical queries (Snowflake, BigQuery, Redshift); schema-on-write
  - Delta Lake -- ACID transactions, schema enforcement, and time travel on top of a data lake; best of both worlds
  - Cloud DS platforms -- AWS SageMaker, Google Vertex AI, Azure ML; managed environments for building and serving models

Module 8: Time Series Analysis

Sessions 24-26

- **Trend, Seasonality & Stationarity:**
  - Time series components: trend (long-term direction), seasonality (repeating patterns), noise (random fluctuation)
  - Decomposition -- additive (Y = T + S + R) or multiplicative (Y = T x S x R); use statsmodels seasonal_decompose()
  - Autocorrelation (ACF) -- correlation of series with its own lagged values; plot ACF to identify MA order
  - Partial autocorrelation (PACF) -- correlation at lag k after removing shorter-lag effects; used to identify AR order
  - Stationarity -- mean and variance constant over time; required for ARIMA; test with ADF (Augmented Dickey-Fuller) test
- **ARIMA Models:**
  - ARIMA(p, d, q) -- AutoRegressive Integrated Moving Average; p: AR lags, d: differencing, q: MA lags
  - AR(p) -- Y_t predicted as a linear combination of p past values; captures persistence and momentum
  - MA(q) -- Y_t predicted as a linear combination of q past error terms; captures short-term shocks
  - Differencing (d) -- take first difference Y_t - Y_{t-1} to make a non-stationary series stationary
  - Model identification -- use ACF/PACF plots to choose p and q; auto_arima from pmdarima selects automatically
  - SARIMA -- Seasonal ARIMA; adds seasonal AR, MA, and differencing terms for strongly seasonal data

Week 6 | Sessions 26-30

Time Series (completion) & Advanced Topics

Mon	Tue	Wed	Thu	Fri
S26	S27	S28	S29	S30

Time Series III -- forecasting techniques; Prophet; LSTM for TS; evaluation metrics (RMSE, MAPE)	Advanced I -- feature selection; filter, wrapper, embedded methods; VIF; multicollinearity	Advanced II -- dimensionality reduction; PCA, t-SNE, UMAP; autoencoders	Advanced III -- anomaly detection; isolation forest, LOF; point, contextual & collective anomalies	Advanced IV -- recommendation systems; A/B testing; experimental design

Module 8: Time Series -- continued

Session 26

- **Forecasting Techniques:**
  - Exponential smoothing -- weight recent observations more; Simple (no trend/seasonality), Holt-Winters (both)
  - Prophet (Meta) -- additive model with trend + seasonality + holidays; robust to missing data; easy to use
  - LSTM for time series -- learns long-term dependencies in sequences; useful for complex non-linear patterns
  - Forecast evaluation -- RMSE and MAE are scale-dependent; MAPE and SMAPE are scale-independent; use both
  - Walk-forward validation -- train up to time t, predict t+1, advance t; the most realistic evaluation strategy for TS

Module 9: Advanced Topics

Sessions 27-30

- **Feature Selection:**
  - Why feature selection: remove irrelevant/redundant features; reduce overfitting; improve speed and interpretability
  - Filter methods -- use statistics independent of the model: chi-square, ANOVA F-test, mutual information; fast
  - Wrapper methods -- use the model to evaluate feature subsets; RFE (Recursive Feature Elimination); computationally expensive
  - Embedded methods -- selection during training; Lasso zeros out irrelevant features; tree feature importance scores
  - Variance Inflation Factor (VIF) -- detect multicollinearity between features; VIF > 10 is problematic; drop or combine
- **Dimensionality Reduction:**
  - Curse of dimensionality -- as features grow, data becomes sparse; distances lose meaning; models need more data
  - PCA -- linear projection onto principal components; maximises variance; unsupervised; use for speed or visualisation
  - t-SNE -- non-linear; preserves local structure; best for 2D/3D visualisation of clusters; not for modelling directly
  - UMAP -- faster than t-SNE; preserves both local and global structure; scalable; increasingly preferred for visualisation
  - Autoencoders -- neural network that learns a compressed bottleneck representation; captures complex non-linear structure
- **Anomaly Detection:**
  - Point anomalies -- single data points that deviate significantly from normal; fraud detection, sensor failure
  - Contextual anomalies -- anomalous only in context (30C is normal in summer, anomalous in winter time series)
  - Collective anomalies -- a sequence is collectively anomalous even if no single point looks unusual alone
  - Isolation Forest -- randomly partitions data; anomalies require fewer splits to isolate; efficient for high dimensions
  - Local Outlier Factor (LOF) -- compares local density of each point to its neighbours; finds outliers in varying density regions
- **Recommendation Systems & A/B Testing:**
  - Collaborative filtering -- recommend based on similar users (user-based) or similar items (item-based); matrix factorisation (SVD)
  - Content-based filtering -- recommend based on item features; TF-IDF for text; cosine similarity between item profiles
  - Cold start problem -- new users/items have no history; hybrid systems (collab + content-based) mitigate this
  - A/B testing -- randomised controlled experiment; control vs treatment; measure the causal effect of a change on a metric
  - A/B test analysis: define metric, compute required sample size (power analysis), run experiment, test for significance

Week 7 | Sessions 31-35

Deep Learning for Data Science & MLOps

Mon	Tue	Wed	Thu	Fri
S31	S32	S33	S34	S35
Deep Learning I -- neural networks; forward/backprop; activations; loss functions; Keras intro	Deep Learning II -- CNNs; conv layers, pooling, transfer learning; image classification hands-on	Deep Learning III -- RNNs, LSTMs, GRUs; sequence data; TS forecasting & text classification	MLOps I -- model deployment; FastAPI; Docker; serialisation; cloud deployment	MLOps II -- monitoring; data drift; MLflow; model versioning; CI/CD for ML

Module 10: Deep Learning for Data Science

Sessions 31-33

- **Neural Networks Overview:**
  - Neural network: layers of neurons; each computes Z = W*X + b, then applies non-linear activation A = f(Z)
  - Forward pass -- input flows through hidden layers to output; each layer extracts increasingly abstract features
  - Backpropagation -- chain rule computes gradients of loss w.r.t. all weights; propagated from output to input
  - Activation functions -- ReLU (most common for hidden layers), Sigmoid (binary output), Softmax (multi-class)
  - Regularisation -- Dropout (randomly zero out neurons during training), L2 weight decay; prevent overfitting
  - Keras with TensorFlow -- model = Sequential(); model.add(Dense()); model.compile(); model.fit(); quick prototyping
- **CNN Basics:**
  - CNNs exploit spatial structure in images; translation-invariant feature detectors via learned filters (kernels)
  - Convolution layer -- kernel slides across input; each filter detects a specific pattern (edges, textures, shapes)
  - Pooling layer -- max pooling reduces spatial dimensions; reduces computation and adds spatial invariance
  - Flatten + Dense layers -- after conv/pool, flatten to 1D vector; dense layers classify based on extracted features
  - Transfer learning -- use pre-trained CNN (ResNet, EfficientNet) as feature extractor; fine-tune on small dataset
  - Applications in DS -- image classification, document scanning, satellite imagery analysis, medical imaging
- **RNN Basics & Applications in Data Science:**
  - RNNs process sequences by maintaining a hidden state that carries information forward across timesteps
  - Vanishing gradient problem -- gradients shrink as they propagate back through many timesteps; early context is forgotten
  - LSTMs (Long Short-Term Memory) -- gates (forget, input, output) solve vanishing gradient; remember long-range dependencies
  - GRUs (Gated Recurrent Units) -- simpler than LSTM; similar performance; fewer parameters; often preferred in practice
  - DS applications: time series forecasting with LSTM; text classification; sentiment analysis; anomaly detection in sequences

Module 11: MLOps & Deployment

Sessions 34-35

- **Model Deployment & APIs:**
  - MLOps = DevOps practices applied to ML; automate the lifecycle from development to production and monitoring
  - FastAPI -- fast async Python framework; define a /predict endpoint; accept JSON input, return JSON prediction
  - Model serialisation -- pickle or joblib for sklearn; ONNX for cross-framework; save model once, serve anywhere
  - Docker -- containerise the model server; Dockerfile installs dependencies and copies the model; reproducible everywhere
  - Cloud deployment -- AWS SageMaker endpoints, Google Cloud Run, Azure ML; managed, scalable, pay-per-use
- **Monitoring, Versioning & CI/CD for ML:**
  - Data drift -- input feature distributions shift over time; model trained on old data degrades in production
  - Model drift -- accuracy drops even if data distributions are stable; monitor business metrics continuously
  - MLflow -- log experiments (parameters, metrics, artifacts); compare runs; model registry for version management
  - Model versioning stages -- Staging (tested), Production (live), Archived; controlled promotion with approvals
  - CI/CD for ML -- GitHub Actions: run tests -> train model -> evaluate -> if metrics pass, deploy; automate the full pipeline

Week 8 | Sessions 36-40

Business Communication & Projects -- Capstone Kickoff

Mon	Tue	Wed	Thu	Fri
S36	**S37**	**S38**	**S39**	**S40**
Business & Comm I -- business problem framing; MECE; stakeholder mapping; success metrics	Business & Comm II -- data storytelling; Pyramid Principle; narrative structure; reporting	Business & Comm III -- visualisation for stakeholders; Power BI basics; decision making with data	Projects I -- end-to-end DS project; Kaggle competition strategy; documentation best practices	Projects II -- industry case studies; portfolio building; capstone kickoff & team formation

Module 12: Business & Communication

Sessions 36-38

- **Business Problem Framing:**
  - Translate business questions into DS problems: "increase revenue" -> "predict churn" -> "binary classification"
  - MECE framework (Mutually Exclusive, Collectively Exhaustive) -- decompose problems without gaps or overlaps
  - Success metric alignment -- agree upfront on the metric; precision vs recall trade-off depends on cost of each error type
  - Stakeholder mapping -- identify who owns the problem, who will use the model, who approves deployment
  - Scope and feasibility -- what data is available, what timeline is realistic, what accuracy is "good enough" to act on
- **Data Storytelling:**
  - Pyramid Principle -- start with the conclusion; support with arguments; each argument backed by specific data
  - Audience-first thinking -- technical audience wants methodology; business audience wants the so-what and next action
  - The narrative arc: context (here is the situation) -> complication (here is the problem) -> resolution (here is what to do)
  - Chart selection: one clear chart per insight; remove chart junk; label data points not just axes
  - Common mistakes: too many slides, too much data, misleading scales, no clear takeaway; always less is more
- **Visualisation for Stakeholders & Reporting:**
  - Executive dashboards -- 3-5 KPIs on one page; traffic light status; trend over time; drill-down capability
  - Power BI, Tableau, Looker -- BI tools for non-technical stakeholders; self-service analytics; no code required
  - Streamlit reports -- Python-based; code and charts in one app; shareable as a web link without BI tools
  - Report structure: executive summary (1 page) -> methodology -> findings -> recommendations -> technical appendix
  - Automated reporting -- schedule with Airflow or cron; parameterise SQL queries; stakeholders always see fresh data
- **Decision Making with Data:**
  - Evidence-based decisions -- replace gut instinct with data; quantify uncertainty; never present a point estimate alone
  - Decision trees for business -- map outcomes and probabilities; expected value = sum(probability x outcome) for each path
  - Causal vs correlational -- correlation does not imply causation; need a randomised experiment or causal inference for attribution
  - Communicating uncertainty -- always include confidence intervals or ranges; stakeholders need to know the limits of your model
  - DS in product and strategy -- A/B testing product features, pricing optimisation, market basket analysis, customer segmentation

Module 13: Projects & Case Studies

Sessions 39-40

- **End-to-End DS Project & Kaggle:**
  - End-to-end structure: problem definition -> data collection -> EDA -> cleaning -> feature engineering -> model -> evaluate -> present
  - Kaggle competitions -- structured problem, clean dataset, public leaderboard; great for learning and portfolio evidence
  - Kaggle winning strategies: start with EDA notebooks, ensemble models (XGBoost + LightGBM), aggressive feature engineering

Project documentation -- README with problem statement, approach, results, and how to reproduce; data science report alongside code

Code quality -- modular functions with docstrings; reproducible environment with requirements.txt; no hardcoded paths

- **Industry Case Studies & Portfolio Building:**

Healthcare DS: patient readmission prediction, medical image classification; data privacy and interpretability are paramount

Finance DS: fraud detection, credit scoring, customer churn; regulatory constraints require explainable models

Retail DS: demand forecasting, price optimisation, recommendation systems; direct and measurable revenue impact

Portfolio building -- 3-5 end-to-end projects on GitHub; diverse domains; each with a clear README and reproducible code

Real-world career paths -- Kaggle competitions, open dataset contributions, DS blogging; public work opens doors

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)

Sessions 41-45

- **End-to-end Data Science project delivery:**

Teams build a complete DS project -- problem definition, EDA, model, and deployed Streamlit dashboard with insights

All work on the organisation GitHub repo -- PRs used for mentor review and integration throughout the capstone

Mentor-led doubt-solving sessions (S41 and S42) -- feedback delivered via PR comments on code and analysis

Mid-capstone checkpoint -- progress review; mentor merges or requests changes; redirect early if needed

- **Final Presentation (Session 44):**

10-minute team demo + 5-minute Q&A

Present: business problem, data used, EDA insights, model choice and performance, dashboard demo, recommendations

Peer review -- structured feedback form for cross-team assessment

Certificate of Completion issued upon successful delivery

Resume, Portfolio & Career Readiness Workshop (Session 43)

Session 43

- **Data Science Resume Structure:**

One-page format for 0-3 years experience; two-page maximum for senior roles

Skills section -- Python, pandas, NumPy, scikit-learn, matplotlib/seaborn, SQL, Spark (basic), Streamlit, MLflow, Docker

Projects section -- problem, dataset, approach, model used, metric achieved, business impact (quantified where possible)

ATS optimisation -- keyword alignment with DS job descriptions (EDA, feature engineering, model evaluation, data pipeline)

- **GitHub Portfolio:**

By this point: 9 weeks of daily PRs on the org repo -- a live, reviewable data science learning track record

Capstone repo under the org -- deployed Streamlit app with a live demo link and a well-written README

Personal profile README -- pinned repos, project summaries, tech stack, links to Kaggle profile and demos

README structure for each project -- problem, data source, EDA highlights, model, results, how to run

Contribution graph -- consistent daily activity signals discipline and reliability to recruiters

- **LinkedIn Optimisation:**

  Headline -- "Data Scientist | Python | Machine Learning | SQL | EDA | scikit-learn" style

  About section -- 3-5 sentences: background, what kinds of problems you solve, key tools and what you are seeking

  Featured section -- link to deployed Streamlit app, org GitHub repo, or a Kaggle notebook

  Skills and endorsements -- request endorsements from cohort peers for Python, Data Science, Machine Learning, SQL

- **Interview Guidance:**

  Technical rounds -- explain the DS lifecycle, describe how you would handle a given modelling problem end-to-end

  Statistics questions -- p-values, confidence intervals, Type I/II errors, CLT; these come up in every DS interview

  ML questions -- explain overfitting, how you select features, when to use which algorithm, how you evaluate models

  Coding rounds -- Python with pandas and scikit-learn; clean, efficient data manipulation and model training

  Case study rounds -- given a business problem, walk through how you would approach it with data; structure your answer