AI/ML for Data Scientists

Modern Approaches and Deployment

3-Day Course | AI Elevate


Data Preparation


Deep Learning


MLOps


Responsible AI

Course Overview

Day 1

Foundations & Data

Day 2

Modern ML & Deep Learning

Day 3

Deployment & Ethics

Target Audience

Data Scientists, Business Analysts, and ML Engineers who want to master modern AI/ML techniques and deploy production-ready models responsibly.

Day 1

Foundations & Data Preparation

Building the Foundation for ML Success

Day 1

The Modern ML Landscape

$200B+
Global AI Market 2025

77%
Companies Using AI

10M+
Data Scientists Worldwide

87%
ML Projects Don't Deploy

The Deployment Gap

Most ML projects fail to reach production. This course focuses on bridging that gap with practical skills in data preparation, modern architectures, and responsible deployment.

Day 1

Types of Machine Learning

Supervised Learning

Learn from labeled data to predict outcomes.

- Classification
- Regression
- Ranking

Examples: Spam detection, price prediction, image classification

Unsupervised Learning

Find patterns in unlabeled data.

- Clustering
- Dimensionality Reduction
- Anomaly Detection

Examples: Customer segmentation, fraud detection

Reinforcement Learning

Learn through trial and error with rewards.

- Policy Learning
- Value Functions
- Model-Based RL

Examples: Game AI, robotics, recommendation systems

Day 1

The ML Pipeline

Time Allocation Reality

80% of time: Data collection, cleaning, feature engineering

20% of time: Model training, evaluation, deployment

Day 1

Data Preprocessing

Data Cleaning

- **Missing values:** Imputation, deletion, prediction
- **Outliers:** Detection and handling
- **Duplicates:** Identification and removal
- **Inconsistencies:** Standardization

Data Transformation

- **Scaling:** MinMax, Standard, Robust
- **Encoding:** One-hot, Label, Target
- **Binning:** Equal-width, Equal-frequency

```
import pandas as pd from sklearn.preprocessing import StandardScaler from sklearn.impute import SimpleImputer # Handle missing values imputer = SimpleImputer(strategy='median') df['age'] = imputer.fit_transform(df[['age']]) # Scale numeric features scaler = StandardScaler() df['age_scaled'] = scaler.fit_transform(df[['age']]) # One-hot encode categories df = pd.get_dummies(df, columns=['category'])
```

- **Log/Power:** For skewed distributions

Day 1

Feature Engineering

The art of creating informative features from raw data.

Temporal Features

- Day of week
- Month/ Season
- Hour of day
- Time since event
- Rolling statistics

Text Features

- TF-IDF
- Word embeddings
- N-grams
- Sentiment scores
- Named entities

Numeric Features

- Ratios
- Differences
- Aggregations
- Polynomial features
- Binning

Categorical Features

- Target encoding
- Frequency encoding
- Embedding layers
- Category combinations
- Hierarchies

Pro Tip: Domain Knowledge is Key

The best features often come from understanding the business problem deeply. Collaborate with domain experts to uncover meaningful patterns in your data.

Day 1

Model Selection Guide

Algorithm	Best For	Pros	Cons
Linear/Logistic Regression	Baseline, interpretability	Fast, interpretable, good baseline	Limited to linear relationships
Random Forest	Tabular data, feature importance	Robust, handles non-linear	Memory intensive, slower inference
XGBoost/LightGBM	Competitions, tabular data	State-of-the-art accuracy	Prone to overfitting, tuning needed
Neural Networks	Images, text, complex patterns	Learns representations	Data hungry, compute intensive
SVMs	Small datasets, high dimensions	Effective in high dimensions	Doesn't scale well

Decision Framework

Start simple → Add complexity as needed → Validate rigorously

Day 1

Evaluation Metrics

Classification Metrics

	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

$$\text{Precision} = TP / (TP + FP)$$
$$\text{Recall} = TP / (TP + FN)$$
$$F1 = 2 \times (P \times R) / (P + R)$$

Regression Metrics

MAE (Mean Absolute Error)

Average absolute difference. Robust to outliers.

MSE / RMSE

Squared errors. Penalizes large errors more.

R^2 (Coefficient of Determination)

Proportion of variance explained. Range: 0-1.

MAPE

Percentage error. Scale-independent.

Day 1

Cross-Validation Strategies

K-Fold CV

Split data into k folds, train on k-1, test on 1. Repeat k times.

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score( model, X, y, cv=5 )
```

Stratified K-Fold

Preserves class distribution in each fold. Essential for imbalanced data.

```
from sklearn.model_selection import StratifiedKFold skf = StratifiedKFold(n_splits=5)
```

Time Series Split

For temporal data. Train on past, test on future. No data leakage.

```
from sklearn.model_selection import TimeSeriesSplit tscv = TimeSeriesSplit(n_splits=5)
```

Data Leakage Warning

Never use test data for any preprocessing decisions. Fit scalers, imputers, and encoders only on training data!

Day 1

Essential Tools & Libraries

- Python
- NumPy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn
- PyTorch
- TensorFlow
- XGBoost
- LightGBM
- Hugging Face
- Jupyter
- MLflow
- Docker
- Cloud ML

Day 2

Deep Learning & Modern Architectures

Neural Networks to Transformers

Day 2

Neural Networks Fundamentals

The Artificial Neuron

$$y = f(\Sigma(w_i x_i) + b)$$

Common Activation Functions

ReLU: max(0, x)

Most common. Fast, avoids vanishing gradient.

Sigmoid: $1/(1+e^{-x})$

Output 0-1. Used for binary classification.

Softmax

Multi-class probabilities. Outputs sum to 1.

Tanh: $(e^x-e^{-x})/(e^x+e^{-x})$

Output -1 to 1. Zero-centered.

Day 2

Deep Learning Architectures

CNNs

Convolutional Neural Networks

- Image classification
- Object detection
- Semantic segmentation
- Medical imaging

Key: Convolution, Pooling, Feature maps

RNNs / LSTMs

Recurrent Neural Networks

- Sequence modeling
- Time series
- Speech recognition
- Language modeling

Key: Memory cells, Hidden states

Transformers

Attention Is All You Need

- NLP (BERT, GPT)
- Vision (ViT)
- Multimodal (CLIP)
- Foundation models

Key: Self-attention, Parallelization

Day 2

CNN Architecture Deep Dive

```
import torch.nn as nn class SimpleCNN(nn.Module): def __init__(self, num_classes=10):
super().__init__() self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1) self.conv2 =
```

```
nn.Conv2d(32, 64, kernel_size=3, padding=1) self.pool = nn.MaxPool2d(2, 2) self.fc =
nn.Linear(64 * 56 * 56, num_classes)
```

Day 2

Transformers & Self-Attention

Self-Attention Mechanism

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T/\sqrt{d_k})V$$

Why Transformers Win

- Parallel processing (vs sequential RNNs)
- Long-range dependencies

Popular Transformer Models

BERT

Bidirectional encoder. Great for classification, NER, Q&A.

GPT Family

Autoregressive decoder. Text generation, chat, code.

T5 / BART

Encoder-decoder. Summarization, translation.

- Transfer learning ready

Vision Transformer (ViT)

Transformers for images. Competitive with CNNs.

Day 2

Transfer Learning

Leverage pre-trained models for your specific task.

1. **Pre-trained Model**
   Trained on large dataset (ImageNet, Wikipedia)
2. **Freeze Base Layers**
   Keep learned representations intact
3. **Add Custom Head**
   Task-specific output layers
4. **Fine-tune**
   Train on your data (optionally unfreeze)

```
from transformers import AutoModelForSequenceClassification # Load pre-trained model model = AutoModelForSequenceClassification.from_pretrained( "bert-base-uncased", num_labels=2 ) # Freeze base layers for param in model.bert.parameters(): param.requires_grad = False # Only train classification head optimizer = AdamW(model.classifier.parameters())
```

Benefits

- Less data required
- Faster training
- Better generalization

Day 2

AutoML & Automated Feature Engineering

What AutoML Automates

Feature Engineering

Automatic creation, selection, and transformation of features.

Model Selection

Search across algorithm families to find best fit.

Popular AutoML Tools

Auto-sklearn	Scikit-learn wrapper
H2O AutoML	Enterprise-ready
Google AutoML	Cloud-based, vision/NLP
Azure AutoML	Azure integration
TPOT	Genetic programming
Optuna	Hyperparameter optimization

⚙ Hyperparameter Tuning

Bayesian optimization, grid/random search.

Architecture Search

Neural architecture search (NAS) for deep learning.

AutoML Doesn't Replace You

It accelerates experimentation but requires human oversight for data quality, business logic, and deployment decisions.

Day 2

Multimodal AI

Models that understand and generate across multiple modalities.

Text

Natural language understanding and generation

Images

Computer vision, generation, editing

Audio

Speech recognition, synthesis, music

Video

Understanding, generation, editing

CLIP (OpenAI)

Text-image understanding. Zero-shot classification.

GPT-4V / Gemini

Vision + language. Image understanding with text.

Stable Diffusion / DALL-E

Text-to-image generation.

Day 3

MLOps & Responsible AI

From Development to Production

Day 3

MLOps: Machine Learning Operations

MLOps bridges the gap between ML development and production deployment.

87%
ML projects never reach production

55%
Companies lack MLOps practices

MLOps = DevOps + ML

Continuous integration, delivery, and training for ML systems.

MLOps Capabilities

Experiment Tracking

Log parameters, metrics, artifacts

Model Registry

Version and manage trained models

Model Serving

Deploy models as APIs/services

Monitoring

Track performance and data drift

Automation

CI/CD pipelines for ML

Day 3

Experiment Tracking with MLflow

```
import mlflow # Start experiment
mlflow.set_experiment("customer_churn")
with mlflow.start_run(): # Log parameters
mlflow.log_param("learning_rate", 0.01)
mlflow.log_param("max_depth", 5) # Train
model model = train_model(X_train, y_train)
# Log metrics mlflow.log_metric("accuracy",
0.92) mlflow.log_metric("f1_score", 0.89) #
Log model mlflow.sklearn.log_model(model,
"model")
```

MLflow Components

Tracking

Log and query experiments: parameters, metrics, artifacts.

Projects

Package ML code for reproducibility.

Model Registry

Version models, stage transitions (dev → staging → prod).

Model Serving

Deploy models as REST APIs.

Day 3

Model Deployment Strategies

REST API

Synchronous inference via HTTP endpoints.

- Flask, FastAPI
- Real-time predictions
- Request/response pattern

Best for: Low latency, interactive apps

Batch Processing

Process large datasets offline.

- Apache Spark, Airflow
- Scheduled jobs
- High throughput

Best for: Reports, bulk scoring

Edge Deployment

Run models on devices.

- ONNX, TensorFlow Lite
- Mobile, IoT devices
- Offline capable

Best for: Privacy, low latency

```
# FastAPI deployment example from fastapi import FastAPI import joblib app = FastAPI() model = joblib.load("model.pkl") @app.post("/predict") async def predict(features: List[float]): prediction = model.predict([features]) return {"prediction": prediction.tolist()}
```

Day 3

Model Monitoring & Maintenance

What to Monitor

Data Drift

Input data distribution changes over time. Compare to training data.

Concept Drift

Relationship between inputs and outputs changes.

Performance Metrics

Accuracy, latency, throughput, error rates.

When to Retrain

Triggers for Retraining

- Performance drops below threshold
- Significant data drift detected
- New labeled data available
- Scheduled retraining (weekly/monthly)
- Business requirements change

Monitoring Tools

- Evidently AI
- Whylogs
- Prometheus + Grafana

Operational Metrics

CPU/memory usage, request volume, errors.

- AWS SageMaker Monitor

Responsible AI

Building Trustworthy AI Systems

Day 3

Pillars of Responsible AI

Fairness

- Equal treatment across groups
- Bias detection and mitigation
- Demographic parity
- Equal opportunity

Transparency

- Explainable predictions
- Model documentation
- Decision traceability
- Clear communication

Privacy & Security

- Data protection
- Differential privacy
- Secure inference
- Access controls

Accountability

- Human oversight
- Audit trails
- Governance frameworks
- Impact assessment

Day 3

Bias Detection & Mitigation

Sources of Bias

Data Bias

Training data doesn't represent the real world. Historical biases encoded in data.

Algorithm Bias

Model amplifies existing patterns. Proxy variables encode protected attributes.

Mitigation Strategies

Pre-processing

Reweight/resample data, remove sensitive features.

In-processing

Fairness constraints in training, adversarial debiasing.

Post-processing

Adjust thresholds per group, calibration.

Selection Bias

Feedback loops. Self-fulfilling predictions.

Tools

AI Fairness 360, Fairlearn, What-If Tool

Day 3

Model Explainability

SHAP Values

Shapley values from game theory. Feature contributions to each prediction.

```
import shap explainer =
shap.Explainer(model)
shap_values = explainer(X)
shap.plots.waterfall(shap_values[0])
```

LIME

Local Interpretable Model-agnostic Explanations. Approximates locally.

```
from lime import
lime_tabular explainer =
lime_tabular.LimeTabularExplainer(X)
exp =
explainer.explain_instance(x,
model.predict_proba)
```

Feature Importance

Built-in for tree models. Permutation importance for any model.

```
from
sklearn.inspection
import
permutation_importance
result =
permutation_importance(model,
X, y)
```

Global vs. Local Explanations

Global: Overall model behavior, feature importance rankings

Local: Why this specific prediction was made

Day 3

Real-World Case Studies

✅ Success: Fraud Detection

Challenge: $10M+ annual fraud losses

Solution: Ensemble model with real-time scoring

Result: 60% reduction in fraud, <1% false positives

Key: Feature engineering on transaction patterns, continuous retraining

✅ Success: Predictive Maintenance

Challenge: Unplanned equipment downtime

Solution: LSTM on sensor time series

❌ Failure: Hiring Algorithm

Issue: Model trained on historical hiring data

Problem: Perpetuated gender bias from past decisions

Lesson: Audit for bias before deployment

Fix: Fairness constraints, diverse training data

❌ Failure: Healthcare Risk Model

Issue: Used healthcare costs as proxy for health

Problem: Underserved groups had lower costs, marked as "healthier"

Result: 35% reduction in downtime, $5M saved

Key: Edge deployment, explainable alerts

Lesson: Validate target variable reflects true goal

Fix: Use direct health measures, not cost proxies

Key Takeaways

Day 1

Foundations Matter

- 80% of time is data prep
- Feature engineering is an art
- Start simple, add complexity
- Cross-validation is essential
- Avoid data leakage

Day 2

Modern Techniques

- Deep learning for unstructured data
- Transformers revolutionized NLP
- Transfer learning saves time/data
- AutoML accelerates experimentation
- Multimodal is the future

Day 3

Production Reality

- MLOps bridges the deployment gap
- Monitor for drift continuously
- Responsible AI is non-negotiable
- Explainability builds trust
- Learn from failures

Your Next Steps

1. Pick a real problem → 2. Start with good data practices → 3. Experiment rapidly → 4. Deploy responsibly → 5. Monitor and iterate

Thank You!

Go Build Amazing AI Systems

Questions?
training@ai-elevate.ai

More Courses
ai-elevate.ai/courses

Resources
Labs & Notebooks