Step 1 — Statistics Foundations

Goal: Build the theoretical foundation to understand models.
Core topics:

- Descriptive statistics: mean, variance, std, percentiles
- Probability distributions: normal, binomial, Poisson, exponential
- Hypothesis testing:
  - χ^2 test (Chi-Square)
  - t-test, ANOVA
  - Pearson/Spearman correlation
- Bias-variance tradeoff
- VC dimension, PAC learning
- Information theory basics: entropy, cross-entropy, KL divergence

Tools:

- No special library needed — use **NumPy/SciPy** for calculations

---

Step 2 — Core Libraries

Goal: Learn the main Python tools for data science & ML.
Libraries & Focus:

- **NumPy** → Arrays, math ops, statistics
- **Pandas** → Data handling, cleaning, joins, grouping
- **Scikit-learn** → Preprocessing, ML models, model evaluation
- **PyTorch** → Deep learning, model training, saving/loading models
- *(Optional)* Matplotlib/Seaborn for visualization

Step 3 — ETL, Encoding, and Feature Selection

Goal: Learn how to get raw data ready for models.
ETL Process:

- **Extract** → from CSV, SQL, APIs
- **Transform** → handle missing values, scaling, normalization
- **Load** → into model pipelines

Encoding:

- One-hot encoding
- Label encoding

Feature Selection:

- Filter methods: correlation, χ^2
- Wrapper methods: recursive feature elimination
- Embedded methods: Lasso, tree-based feature importance

Library: Pandas + Scikit-learn (`preprocessing`, `feature_selection`)

Step 4 — Machine Learning Models

Goal: Understand core ML algorithms and how to test them.
Models:

- Linear Regression (single & multiple)
- Logistic Regression
- Decision Tree
- Random Forest
- SVM
- K-Means (unsupervised)

Model Testing (Scikit-learn):

- **Metrics:**
  - Regression → MAE, MSE, RMSE, R^2
  - Classification → Accuracy, Precision, Recall, F1, ROC-AUC

- **Validation:**
    - Train-test split
    - K-fold cross-validation
    - GridSearchCV / RandomizedSearchCV for hyperparameter tuning

---

Step 5 — Deep Learning Models

Goal: Build & evaluate modern neural network architectures.
Models (PyTorch):

- ANN (feedforward network)
- CNN (images)
- RNN/LSTM/GRU (sequence data)
- Transformers (basic)

Testing Deep Learning Models:

- **Loss functions:** CrossEntropyLoss, MSELoss, etc.
- **Evaluation metrics:** (same as ML where applicable)
- Train-validation-test split
- Early stopping, learning curves

Step 6 — Generative AI

Goal: Learn transformer-based GenAI models & deployment.
Topics:

- **Transformers architecture:** Encoder, Decoder, Encoder-Decoder
- **Self-attention & multi-head attention**

Models:

- BERT (encoder)
- GPT (decoder)
- T5 / BART (encoder-decoder)
- LLaMA, Falcon, Mistral

Hugging Face:

- `transformers` for loading & fine-tuning
- `datasets` for training data

Deployment:

- PyTorch model save/load
- Hugging Face Spaces (Gradio UI)
- REST API with FastAPI

Step 7 — Deployment in Real Life

Goal: Deploy ML, DL, and GenAI models so others can use them.
Options:
Local API:

- Flask / FastAPI for serving models
- `pickle` for ML models
- `torch.save` for DL/GenAI models

Web Apps:

- Streamlit / Gradio
- Hugging Face Spaces