**🧠 Machine Learning Cheat Sheet**

**📚 1. Core Concepts**

* **Types of ML**
  + *Supervised Learning*: Labeled data (e.g., regression, classification)
  + *Unsupervised Learning*: Unlabeled data (e.g., clustering, dimensionality reduction)
  + *Semi-supervised*: Mix of labeled and unlabeled
  + *Reinforcement Learning*: Agent learns via rewards/punishments
* **Key Terminology**
  + *Feature*: Input variable
  + *Label*: Output variable (target)
  + *Model*: Mathematical representation of a process
  + *Training*: Learning from data
  + *Inference*: Making predictions
  + *Overfitting*: Model too closely fits training data
  + *Underfitting*: Model too simple to capture patterns

**🛠️ 2. Data Preprocessing**

* **Cleaning**
  + Handle missing values (mean, median, drop)
  + Remove duplicates
  + Fix inconsistent formats
* **Encoding**
  + Label Encoding
  + One-Hot Encoding
* **Scaling**
  + StandardScaler (mean=0, std=1)
  + MinMaxScaler (0 to 1)
* **Feature Engineering**
  + Polynomial features
  + Interaction terms
  + Domain-specific transformations
* **Dimensionality Reduction**
  + PCA (Principal Component Analysis)
  + t-SNE (for visualization)

**📊 3. Exploratory Data Analysis (EDA)**

* **Visualization Tools**
  + matplotlib, seaborn, plotly
* **Common Plots**
  + Histograms, Boxplots, Pairplots
  + Correlation heatmaps
* **Statistical Summary**
  + Mean, median, std, skewness, kurtosis

**🤖 4. Model Selection**

| **Task** | **Algorithms** |
| --- | --- |
| Regression | Linear Regression, SVR, XGBoost |
| Classification | Logistic Regression, Random Forest, SVM, KNN, XGBoost |
| Clustering | K-Means, DBSCAN, Hierarchical |
| Dim. Reduction | PCA, LDA |

**🧪 5. Model Evaluation**

* **Metrics**
  + *Regression*: MAE, MSE, RMSE, R²
  + *Classification*: Accuracy, Precision, Recall, F1, ROC-AUC
* **Validation Techniques**
  + Train/Test Split
  + K-Fold Cross Validation
  + Stratified Sampling

**🧮 6. Advanced Techniques**

* **Ensemble Methods**
  + Bagging (Random Forest)
  + Boosting (AdaBoost, Gradient Boosting, XGBoost, LightGBM)
  + Stacking
* **Hyperparameter Tuning**
  + Grid Search
  + Random Search
  + Bayesian Optimization
  + Tools: scikit-learn, Optuna, Ray Tune
* **Feature Selection**
  + Recursive Feature Elimination (RFE)
  + Lasso Regularization
  + Tree-based importance

**🧬 7. Deep Learning (DL)**

* **Frameworks**
  + TensorFlow, Keras, PyTorch
* **Components**
  + Neurons, Layers, Activation Functions
  + Loss Functions (MSE, Cross-Entropy)
  + Optimizers (SGD, Adam)
* **Architectures**
  + CNNs (images)
  + RNNs, LSTMs (sequences)
  + Transformers (NLP)

**🚀 8. Deployment**

* **Serialization**
  + pickle, joblib, ONNX
* **Serving**
  + REST APIs: FastAPI, Flask
  + Model Hosting: AWS SageMaker, Azure ML, GCP AI Platform
* **Monitoring**
  + Drift detection
  + Performance tracking
  + Retraining pipelines

**🧰 9. Tools & Libraries**

| **Category** | **Tools/Libraries** |
| --- | --- |
| Data Handling | pandas, numpy, cuDF |
| Modeling | scikit-learn, xgboost, lightgbm |
| Deep Learning | tensorflow, keras, pytorch |
| Visualization | matplotlib, seaborn, plotly |
| Deployment | flask, fastapi, mlflow |

**📈 10. Workflow Summary**

1. Define problem

2. Collect and clean data

3. Explore and visualize

4. Engineer features

5. Select model

6. Train and validate

7. Tune hyperparameters

8. Evaluate performance

9. Deploy and monitor

**Machine Learning Project Lifecycle Cheat Sheet**

A comprehensive, step-by-step guide to the machine learning project lifecycle. Use this as a checklist and reference for your ML projects.

**📋 Phase 1: Problem Definition & Scoping**

*Before writing a single line of code*

| **Step** | **Key Questions** | **Output** |
| --- | --- | --- |
| **Define Goal** | What business problem are we solving? What are the success metrics? (e.g., reduce churn by 10%) | A clear, measurable objective |
| **Identify ML Solution** | Is this supervised (classification, regression), unsupervised (clustering), or reinforcement learning? | Type of ML problem |
| **Determine Requirements** | What data is needed? How will the model be integrated? (Batch vs. Real-time) | Data sources and deployment constraints |
| **Establish Metrics** | **Business Metric:** Bottom-line impact (e.g., increased revenue) **ML Metric:** Model evaluation (e.g., Accuracy, F1-Score, MAE) | Primary and secondary evaluation metrics |

**📊 Phase 2: Data Collection & Understanding**

*Garbage In, Garbage Out. Know your data.*

| **Step** | **Description** | **Key Tools/Techniques** |
| --- | --- | --- |
| **Gather Data** | Collect data from databases, APIs, files, etc. | SQL, Pandas (pd.read\_csv, pd.read\_sql), APIs |
| **Exploratory Data Analysis (EDA)** | Understand structure, patterns, and quirks of the data | **Pandas Profiling**, df.describe(), df.info() |
| **Visualize Data** | Uncover relationships, distributions, and outliers | **Matplotlib**, **Seaborn** (histograms, box plots, scatter plots, heatmaps) |
| **Check for Imbalance** | For classification, is the target variable distributed evenly? | df['target'].value\_counts(), SMOTE, Undersampling |

**🧹 Phase 3: Data Preprocessing & Cleaning**

*Prepare the data for the algorithm.*

| **Category** | **Task** | **Common Methods/Code** |
| --- | --- | --- |
| **Handling Missing Data** | Decide how to treat NaN values | SimpleImputer (mean, median, most\_frequent), dropna() |
| **Encoding Categorical Data** | Convert text categories to numbers | **Ordinal Encoding:** OrdinalEncoder() **One-Hot Encoding:** OneHotEncoder() |
| **Feature Scaling** | Bring all features to similar scale | **Standardization:** StandardScaler() (mean=0, std=1) **Normalization:** MinMaxScaler() (scales to [0,1] range) |
| **Handling Outliers** | Detect and treat extreme values | Visualization (box plots), IQR method, capping, transformation |
| **Feature Engineering** | Create new features from existing ones | Polynomial features, binning, domain-specific features (e.g., "age\_group" from "age") |

**🧠 Phase 4: Model Training & Selection**

*The core of machine learning.*

| **Step** | **Description** | **Key Tools/Techniques** |
| --- | --- | --- |
| **Train-Test Split** | Split data into training and testing sets | from sklearn.model\_selection import train\_test\_split |
| **Select Models** | Choose candidate algorithms to try | **Classification:** Logistic Regression, Random Forest, XGBoost, SVM **Regression:** Linear Regression, Ridge/Lasso, Random Forest Regressor **Clustering:** K-Means, DBSCAN |
| **Train Models** | Fit models on the training data | model.fit(X\_train, y\_train) |
| **Cross-Validation** | Robust performance assessment across data subsets | from sklearn.model\_selection import cross\_val\_score |
| **Hyperparameter Tuning** | Optimize model parameters for best performance | **GridSearchCV**, **RandomizedSearchCV** |

**📈 Phase 5: Model Evaluation**

*How good is your model really?*

**Classification Metrics**

| **Metric** | **Description** | **Interpretation** |
| --- | --- | --- |
| **Accuracy** | Overall correctness | Higher is better |
| **Precision** | % of positive identifications that were correct | Higher is better |
| **Recall** | % of actual positives identified correctly | Higher is better |
| **F1-Score** | Harmonic mean of Precision & Recall | Higher is better |
| **ROC-AUC** | Model's ability to distinguish classes | Closer to 1 is better |

**Regression Metrics**

| **Metric** | **Description** | **Interpretation** |
| --- | --- | --- |
| **MAE** | Mean Absolute Error - average error | Closer to 0 is better |
| **MSE** | Mean Squared Error - punishes larger errors | Closer to 0 is better |
| **R² Score** | % of variance explained by the model | Closer to 1 is better |

**Clustering Metrics**

| **Metric** | **Description** | **Interpretation** |
| --- | --- | --- |
| **Silhouette Score** | How well-separated clusters are | Higher is better |
| **Inertia** | Sum of squared distances to cluster center | Lower is better |

**🚀 Phase 6: Model Deployment & Monitoring**

*Making your model useful.*

| **Step** | **Description** | **Tools/Concepts** |
| --- | --- | --- |
| **Save Model** | Serialize the trained model to a file | import pickle or import joblib |
| **Create API** | Wrap model in an API endpoint for predictions | **Flask**, **FastAPI**, **Django** |
| **Deploy** | Host model in a production environment | **Cloud (AWS SageMaker, GCP AI Platform, Azure ML), Docker, Kubernetes** |
| **Monitor** | Track performance to detect model drift | **Prometheus, Grafana, Evidently AI** |
| **Retrain** | Set up pipeline for periodic retraining | **Apache Airflow, MLflow, CI/CD pipelines** |

**🔄 Phase 7: Reproducibility & Best Practices**

*Doing things the right way.*

| **Practice** | **Description** | **Tools** |
| --- | --- | --- |
| **Version Control** | Version your code AND your data | **Git, DVC (Data Version Control)** |
| **Experiment Tracking** | Log parameters, metrics, and artifacts for each run | **MLflow, Weights & Biases, Neptune.ai** |
| **Documentation** | Document process, assumptions, and results | Jupyter Notebooks, Markdown, Confluence |
| **Modular Code** | Write reusable functions and scripts | Python scripts, import statements |

**💻 Quick Code Skeleton**

*# 1. Import Libraries*

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

import pickle

*# 2. Load & Explore Data*

df = pd.read\_csv('data.csv')

print(df.head())

print(df.describe())

*# 3. Preprocess Data*

*# ... Handle missing values, encode categories, etc.*

X = df.drop('target', axis=1)

y = df['target']

*# 4. Split Data*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

*# 5. Scale Features (if needed)*

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

*# 6. Train Model*

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

*# 7. Evaluate Model*

y\_pred = model.predict(X\_test)

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")

*# 8. Save Model for Later*

with open('model.pkl', 'wb') as file:

pickle.dump(model, file)

**🎯 Key Principles**

* **Start Simple**: Begin with baseline models before trying complex algorithms
* **Iterate**: ML is an iterative process - expect to go back to previous steps
* **Validate**: Always validate your model on unseen data
* **Monitor**: Models can degrade over time (model drift) - plan for monitoring and retraining

*This cheat sheet provides a high-level roadmap. Each step has deep underlying concepts, but following this structure will ensure you never miss a critical part of the ML process.*