Complete Machine Learning Cheatsheet

Table of Contents

1. Introduction & Core Concepts
2. Types of Machine Learning
3. Machine Learning Workflow
4. Data Preprocessing
5. Supervised Learning Algorithms
6. Unsupervised Learning Algorithms
7. Model Evaluation & Metrics
8. Feature Engineering
9. Model Selection & Hyperparameter Tuning
10. Deep Learning Fundamentals
11. Common Challenges & Solutions
12. Best Practices
13. Tools & Frameworks

Introduction & Core Concepts

What is Machine Learning?

Machine Learning (ML) is a subset of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed. It focuses on developing algorithms that can access data and use it to learn for themselves.

Key Terminology

- **Model**: A mathematical representation of patterns in data
- **Training**: The process of teaching an algorithm to make predictions
- **Features**: Input variables used to make predictions
- **Labels**: Output variables we're trying to predict
- **Instance/Sample**: A single data point
- **Dataset**: Collection of instances
- **Hypothesis**: The function that maps inputs to outputs
- **Parameters**: Internal variables learned during training
- **Hyperparameters**: Configuration settings set before training

Mathematical Foundation

- **Loss Function**: Measures how wrong predictions are (e.g., MSE, Cross-entropy)
- **Optimization**: Process of minimizing the loss function
- **Gradient Descent**: Iterative optimization algorithm
- **Learning Rate**: Step size in gradient descent

Types of Machine Learning

1. Supervised Learning

Learning from labeled data where both input and output are known.

Applications:

- Classification (spam detection, image recognition)
- Regression (price prediction, demand forecasting)

2. Unsupervised Learning

Learning patterns from unlabeled data without predefined outputs.

Applications:

- Clustering (customer segmentation)
- Dimensionality reduction (data visualization)
- Anomaly detection

3. Semi-Supervised Learning

Combines small amounts of labeled data with large amounts of unlabeled data.

4. Reinforcement Learning

Learning through interaction with an environment using rewards and penalties.

Components:

- Agent: The learner/decision maker
- Environment: What the agent interacts with
- State: Current situation
- Action: What the agent can do
- Reward: Feedback from the environment

Machine Learning Workflow

1. Problem Definition

- Identify the business problem
- Define success metrics
- Determine ML feasibility

2. Data Collection

- Gather relevant data
- Ensure data quality and quantity
- Consider data privacy and ethics

3. Data Exploration (EDA)

- Understand data distribution
- Identify patterns and relationships
- Detect anomalies and outliers

4. Data Preprocessing

- Handle missing values
- Encode categorical variables
- Scale/normalize features
- Split data (train/validation/test)

5. Model Selection

- Choose appropriate algorithms
- Consider problem constraints
- Start with simple baselines

6. Model Training

- Fit model to training data
- Monitor training progress
- Prevent overfitting

7. Model Evaluation

- Assess performance on validation set
- Use appropriate metrics

- Compare against baselines

8. Model Deployment

- Integrate into production system
- Monitor performance
- Plan for updates and maintenance

Data Preprocessing

Handling Missing Data

1. **Deletion Methods**:
   - Remove rows with missing values
   - Remove features with high missing percentage
2. **Imputation Methods**:
   - Mean/Median/Mode imputation
   - Forward/Backward fill
   - K-NN imputation
   - Model-based imputation

Feature Scaling

1. **Standardization (Z-score normalization)**:

   ```
   z = (x - μ) / σ
   ```

   - Centers data around 0 with σ = 1
   - Good for normally distributed features
2. **Min-Max Scaling**:

   ```
   x_scaled = (x - min) / (max - min)
   ```

   - Scales features to [0, 1]
   - Sensitive to outliers
3. **Robust Scaling**:
   - Uses median and IQR
   - Less sensitive to outliers

Encoding Categorical Variables

1. **One-Hot Encoding**: Creates binary columns for each category

2. **Label Encoding**: Assigns integer to each category
3. **Target Encoding**: Uses target statistics
4. **Embedding**: Learned dense representations (for high cardinality)

Data Splitting

- **Training Set** (60-80%): For model training
- **Validation Set** (10-20%): For hyperparameter tuning
- **Test Set** (10-20%): For final evaluation
- **Time Series**: Respect temporal order

Supervised Learning Algorithms

Classification Algorithms

1. Logistic Regression

- **Use Case**: Binary/multiclass classification
- **Pros**: Simple, interpretable, probabilistic
- **Cons**: Assumes linear relationships
- **Key Parameters**: C (regularization), penalty type

2. Decision Trees

- **Use Case**: Both classification and regression
- **Pros**: Interpretable, handles non-linearity
- **Cons**: Prone to overfitting
- **Key Parameters**: max_depth, min_samples_split

3. Random Forest

- **Use Case**: Complex non-linear problems
- **Pros**: Reduces overfitting, feature importance
- **Cons**: Less interpretable, computationally expensive
- **Key Parameters**: n_estimators, max_depth

4. Support Vector Machines (SVM)

- **Use Case**: High-dimensional data
- **Pros**: Effective in high dimensions, memory efficient
- **Cons**: Slow on large datasets, needs scaling

- **Key Parameters**: C, kernel, gamma

5. k-Nearest Neighbors (k-NN)

- **Use Case**: Simple classification/regression
- **Pros**: Simple, no training phase
- **Cons**: Slow prediction, sensitive to scale
- **Key Parameters**: k, distance metric

6. Naive Bayes

- **Use Case**: Text classification, spam filtering
- **Pros**: Fast, works with small data
- **Cons**: Assumes feature independence
- **Types**: Gaussian, Multinomial, Bernoulli

7. Gradient Boosting (XGBoost, LightGBM)

- **Use Case**: Competitions, high accuracy needed
- **Pros**: State-of-the-art performance
- **Cons**: Complex, prone to overfitting
- **Key Parameters**: learning_rate, n_estimators, max_depth

Regression Algorithms

1. Linear Regression

- **Equation**: $y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$
- **Assumptions**: Linearity, independence, homoscedasticity, normality
- **Regularization**: Ridge (L2), Lasso (L1), Elastic Net

2. Polynomial Regression

- **Use Case**: Non-linear relationships
- **Risk**: Overfitting with high degrees

3. Regression Trees & Ensembles

- Similar to classification versions
- Uses MSE/MAE instead of entropy/gini

Unsupervised Learning Algorithms

Clustering Algorithms

1. K-Means

- **How it works**: Partitions data into k clusters
- **Pros**: Simple, scalable
- **Cons**: Assumes spherical clusters, needs k
- **Distance Metric**: Euclidean

2. Hierarchical Clustering

- **Types**: Agglomerative (bottom-up), Divisive (top-down)
- **Pros**: No need to specify k, dendrogram visualization
- **Cons**: Computationally expensive

3. DBSCAN

- **How it works**: Density-based clustering
- **Pros**: Finds arbitrary shapes, outlier detection
- **Cons**: Sensitive to parameters
- **Parameters**: eps (radius), min_samples

4. Gaussian Mixture Models (GMM)

- **How it works**: Assumes data from mixture of Gaussians
- **Pros**: Soft clustering, probabilistic
- **Cons**: Sensitive to initialization

Dimensionality Reduction

1. Principal Component Analysis (PCA)

- **Goal**: Find directions of maximum variance
- **Use**: Visualization, noise reduction, compression
- **Linear**: Only captures linear relationships

2. t-SNE

- **Goal**: Preserve local structure in low dimensions
- **Use**: Visualization of high-dimensional data
- **Note**: Non-parametric, computationally expensive

3. UMAP

- **Goal**: Similar to t-SNE but faster
- **Advantages**: Preserves global structure better

4. Autoencoders

- **Type**: Neural network approach
- **Use**: Non-linear dimensionality reduction

Model Evaluation & Metrics

Classification Metrics

1. Accuracy

```
Accuracy = (TP + TN) / (TP + TN + FP + FN)
```

- **When to use**: Balanced classes
- **Limitation**: Misleading for imbalanced data

2. Precision & Recall

```
Precision = TP / (TP + FP)
Recall = TP / (TP + FN)
```

- **Precision**: Of predicted positives, how many are correct?
- **Recall**: Of actual positives, how many did we find?

3. F1-Score

```
F1 = 2 * (Precision * Recall) / (Precision + Recall)
```

- **Use**: Balance between precision and recall

4. ROC-AUC

- **ROC**: Receiver Operating Characteristic curve
- **AUC**: Area Under the Curve
- **Range**: 0.5 (random) to 1.0 (perfect)

5. Confusion Matrix

- Visualizes TP, TN, FP, FN
- Helps identify specific error types

Regression Metrics

1. Mean Squared Error (MSE)

```
MSE = Σ(y_true - y_pred)² / n
```

- **Characteristic**: Penalizes large errors more

2. Root Mean Squared Error (RMSE)

```
RMSE = √MSE
```

- **Advantage**: Same units as target

3. Mean Absolute Error (MAE)

```
MAE = Σ|y_true - y_pred| / n
```

- **Characteristic**: Less sensitive to outliers

4. R^2 (Coefficient of Determination)

```
R² = 1 - (SS_res / SS_tot)
```

- **Range**: -∞ to 1.0
- **Interpretation**: Proportion of variance explained

Cross-Validation

- **k-Fold**: Split data into k parts, train on k-1, test on 1
- **Stratified k-Fold**: Maintains class distribution
- **Time Series**: Use forward chaining
- **Leave-One-Out**: k = number of samples

Feature Engineering

Feature Creation

1. **Polynomial Features**: x^2, x^3, $x_1 * x_2$
2. **Domain-Specific**: Based on business knowledge
3. **Interaction Features**: Combinations of existing features
4. **Binning**: Convert continuous to categorical

5. **Date/Time Features**: Hour, day, month, season

Feature Selection

1. **Filter Methods**:
   - Correlation analysis
   - Chi-square test
   - Mutual information
2. **Wrapper Methods**:
   - Forward selection
   - Backward elimination
   - Recursive Feature Elimination (RFE)
3. **Embedded Methods**:
   - L1 regularization (Lasso)
   - Tree-based feature importance
   - Permutation importance

Feature Extraction

- **PCA**: Linear combinations of features
- **LDA**: Maximizes class separation
- **Feature Hashing**: For high-cardinality categorical
- **Word Embeddings**: For text data

Model Selection & Hyperparameter Tuning

Model Selection Strategies

1. **Start Simple**: Linear models as baseline
2. **Increase Complexity**: Gradually try complex models
3. **Ensemble Methods**: Combine multiple models
4. **Domain Knowledge**: Use appropriate algorithms

Hyperparameter Tuning

1. Grid Search

- **Approach**: Try all combinations
- **Pros**: Thorough
- **Cons**: Computationally expensive

2. Random Search

- **Approach**: Random sampling of parameters
- **Pros**: More efficient than grid search
- **Cons**: May miss optimal combination

3. Bayesian Optimization

- **Approach**: Uses past results to guide search
- **Pros**: Efficient for expensive evaluations
- **Tools**: Optuna, Hyperopt

4. Early Stopping

- Stop training when validation performance plateaus
- Prevents overfitting
- Saves computational resources

Deep Learning Fundamentals

Neural Network Basics

- **Neuron**: Basic unit (weighted sum + activation)
- **Layer Types**: Input, Hidden, Output
- **Forward Propagation**: Computing predictions
- **Backpropagation**: Computing gradients

Activation Functions

1. **ReLU**: f(x) = max(0, x)
2. **Sigmoid**: f(x) = 1/(1 + e^(-x))
3. **Tanh**: f(x) = (e^x - e^(-x))/(e^x + e^(-x))
4. **Softmax**: For multiclass output

Common Architectures

1. **Feedforward Networks**: Basic architecture
2. **Convolutional Neural Networks (CNN)**: For images
3. **Recurrent Neural Networks (RNN)**: For sequences
4. **Transformers**: State-of-the-art for NLP

Training Considerations

- **Batch Size**: Trade-off between stability and speed
- **Learning Rate**: Critical hyperparameter
- **Optimizers**: SGD, Adam, RMSprop
- **Regularization**: Dropout, weight decay, batch normalization

---

Common Challenges & Solutions

1. Overfitting

Symptoms: High training accuracy, low test accuracy **Solutions**:

- More data
- Regularization (L1/L2)
- Dropout
- Early stopping
- Simpler models
- Cross-validation

2. Underfitting

Symptoms: Low training and test accuracy **Solutions**:

- More complex models
- More features
- Less regularization
- Longer training

3. Class Imbalance

Solutions:

- Resampling (over/under-sampling)
- SMOTE
- Class weights
- Appropriate metrics (not accuracy)
- Threshold tuning

4. Data Leakage

Prevention:

- Proper train/test split

- No future information
- Careful feature engineering
- Cross-validation discipline

5. Concept Drift

Detection: Monitor model performance over time **Solutions**:

- Regular retraining
- Online learning
- Ensemble methods
- Drift detection algorithms

Best Practices

1. Data Quality

- Garbage in, garbage out
- Invest in data cleaning
- Document data sources
- Version control data

2. Experimentation

- Track all experiments
- Use consistent evaluation
- Document decisions
- Reproducible results

3. Model Interpretability

- Start with interpretable models
- Use SHAP/LIME for black boxes
- Feature importance analysis
- Partial dependence plots

4. Production Considerations

- Model monitoring
- A/B testing
- Fallback strategies

- Performance optimization
- Security and privacy

5. Ethical Considerations

- Bias detection and mitigation
- Fairness metrics
- Transparency
- Privacy preservation
- Responsible AI practices

Tools & Frameworks

Programming Languages

1. **Python**: Most popular for ML
   - Libraries: scikit-learn, TensorFlow, PyTorch, pandas, numpy
2. **R**: Statistical computing
   - Libraries: caret, randomForest, xgboost
3. **Julia**: High-performance scientific computing

Machine Learning Libraries

1. **scikit-learn**: Classical ML algorithms
2. **XGBoost/LightGBM**: Gradient boosting
3. **TensorFlow/Keras**: Deep learning
4. **PyTorch**: Deep learning (research-friendly)
5. **Hugging Face**: Pre-trained models

Data Processing

1. **pandas**: Data manipulation
2. **numpy**: Numerical computing
3. **Dask**: Parallel computing
4. **Apache Spark**: Big data processing

Visualization

1. **matplotlib/seaborn**: Statistical plots
2. **plotly**: Interactive visualizations
3. **TensorBoard**: Neural network visualization

MLOps Tools

1. **MLflow**: Experiment tracking
2. **Kubeflow**: ML workflows on Kubernetes
3. **DVC**: Data version control
4. **Weights & Biases**: Experiment tracking and visualization

Cloud Platforms

1. **AWS SageMaker**
2. **Google Cloud AI Platform**
3. **Azure Machine Learning**
4. **Databricks**

Quick Reference Formulas

Loss Functions

- **MSE**: $L = (1/n)\Sigma(y - \hat{y})^2$
- **Cross-Entropy**: $L = -\Sigma(y \log(\hat{y}))$
- **Hinge Loss**: $L = \max(0, 1 - y*\hat{y})$

Regularization

- **L1 (Lasso)**: $\lambda\Sigma|w|$
- **L2 (Ridge)**: $\lambda\Sigma w^2$
- **Elastic Net**: $\lambda_1\Sigma|w| + \lambda_2\Sigma w^2$

Distance Metrics

- **Euclidean**: $d = \sqrt{(\Sigma(x - y)^2)}$
- **Manhattan**: $d = \Sigma|x - y|$
- **Cosine**: $d = 1 - (x \cdot y)/(||x||*||y||)$

Information Theory

- **Entropy**: $H = -\Sigma(p \log p)$
- **Gini**: $G = 1 - \Sigma p^2$
- **Information Gain**: $IG = H(parent) - \Sigma(weight * H(child))$

Conclusion

Machine learning is a vast field that continues to evolve rapidly. This cheatsheet covers the fundamental concepts, algorithms, and practices that form the foundation of ML. Remember that successful ML projects require:

1. Clear problem definition
2. Quality data
3. Appropriate algorithm selection
4. Rigorous evaluation
5. Continuous monitoring and improvement

Keep learning, experimenting, and staying updated with the latest developments in this exciting field!