Complete Machine Learning Cheatsheet

Table of Contents

Types of Machine Learning Problems

1. Supervised Learning

Definition: Learning from labeled data to predict outcomes for new data.

Classification

- **Binary Classification**: Two classes (e.g., spam/not spam)
- **Multi-class Classification**: Multiple classes (e.g., digit recognition 0-9)
- **Multi-label Classification**: Multiple labels per instance

Regression

- **Linear Regression**: Predicting continuous values
- **Non-linear Regression**: Complex relationships between variables

2. Unsupervised Learning

Definition: Finding patterns in unlabeled data.

- **Clustering**: Grouping similar data points
- **Dimensionality Reduction**: Reducing feature space
- **Anomaly Detection**: Identifying outliers
- **Association Rules**: Finding relationships between variables

3. Semi-Supervised Learning

Definition: Combination of labeled and unlabeled data.

4. Reinforcement Learning

Definition: Learning through interaction with environment to maximize reward.

Supervised Learning Algorithms

Classification Algorithms

1. Logistic Regression

- **Use Case**: Binary/multi-class classification
- **Pros**: Simple, interpretable, probabilistic output
- **Cons**: Assumes linear relationship
- **Key Parameters**: C (regularization), solver, penalty

2. Decision Trees

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

3. Random Forest

- **Use Case**: Complex classification/regression problems
- **Pros**: Reduces overfitting, handles missing values
- **Cons**: Less interpretable, computationally expensive
- **Key Parameters**: n_estimators, max_depth, min_samples_split

4. Support Vector Machines (SVM)

- **Use Case**: High-dimensional data, non-linear problems
- **Pros**: Effective in high dimensions, memory efficient
- **Cons**: Slow on large datasets, requires feature 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 feature scale
- **Key Parameters**: n_neighbors, weights, metric

6. Naive Bayes

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

7. Gradient Boosting (XGBoost, LightGBM, CatBoost)

- **Use Case**: Competitions, high-performance needs
- **Pros**: High accuracy, handles various data types
- **Cons**: Prone to overfitting, many hyperparameters
- **Key Parameters**: learning_rate, n_estimators, max_depth

Regression Algorithms

1. Linear Regression

- **Use Case**: Linear relationships
- **Pros**: Simple, interpretable
- **Cons**: Assumes linearity
- **Variants**: Ridge, Lasso, Elastic Net

2. Polynomial Regression

- **Use Case**: Non-linear relationships
- **Pros**: Flexible
- **Cons**: Prone to overfitting

3. Support Vector Regression (SVR)

- **Use Case**: Non-linear regression
- **Pros**: Robust to outliers
- **Cons**: Computationally expensive

Unsupervised Learning Algorithms

Clustering Algorithms

1. K-Means

- **Use Case**: Well-separated spherical clusters
- **Pros**: Fast, scalable
- **Cons**: Requires k specification, sensitive to initialization
- **Key Parameters**: n_clusters, init, max_iter

2. Hierarchical Clustering

- **Use Case**: Hierarchical data structure
- **Pros**: No need to specify k, dendrogram visualization
- **Cons**: Computationally expensive
- **Types**: Agglomerative, Divisive

3. DBSCAN

- **Use Case**: Arbitrary shaped clusters, noise handling
- **Pros**: No need to specify k, finds outliers
- **Cons**: Sensitive to parameters
- **Key Parameters**: eps, min_samples

4. Gaussian Mixture Models (GMM)

- **Use Case**: Soft clustering, overlapping clusters
- **Pros**: Probabilistic approach
- **Cons**: Assumes Gaussian distribution

Dimensionality Reduction

1. Principal Component Analysis (PCA)

- **Use Case**: Linear dimensionality reduction
- **Pros**: Fast, well-understood
- **Cons**: Linear only, loses interpretability

2. t-SNE

- **Use Case**: Visualization of high-dimensional data
- **Pros**: Excellent for visualization
- **Cons**: Computationally expensive, not for new data

3. UMAP

- **Use Case**: General dimensionality reduction
- **Pros**: Faster than t-SNE, preserves global structure
- **Cons**: Many hyperparameters

4. Autoencoders

- **Use Case**: Non-linear dimensionality reduction
- **Pros**: Very flexible
- **Cons**: Requires neural network training

Reinforcement Learning

Key Concepts

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

Common Algorithms

1. **Q-Learning**: Value-based method
2. **Deep Q-Network (DQN)**: Q-learning with neural networks
3. **Policy Gradient**: Direct policy optimization
4. **Actor-Critic**: Combines value and policy methods
5. **PPO (Proximal Policy Optimization)**: Stable policy gradient

---

Data Preprocessing Techniques

1. Handling Missing Data

- **Deletion**: Remove rows/columns with missing values
- **Imputation**:
  - Mean/Median/Mode imputation
  - Forward/Backward fill
  - KNN imputation
  - MICE (Multiple Imputation)

2. Feature Scaling

- **Standardization (Z-score)**: Mean=0, SD=1

```
z = (x - μ) / σ
```

- **Min-Max Normalization**: Scale to [0,1]

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

- **Robust Scaling**: Using median and IQR

3. Encoding Categorical Variables

- **Label Encoding**: Ordinal categories
- **One-Hot Encoding**: Nominal categories
- **Target Encoding**: Based on target variable
- **Binary Encoding**: For high cardinality

4. Handling Imbalanced Data

- **Oversampling**: SMOTE, ADASYN
- **Undersampling**: Random, Tomek Links
- **Class Weights**: Adjust algorithm weights
- **Ensemble Methods**: BalancedRandomForest

5. Outlier Detection & Treatment

- **Statistical Methods**: Z-score, IQR
- **Isolation Forest**
- **Local Outlier Factor (LOF)**
- **Treatment**: Cap, transform, or remove

Feature Engineering

1. Feature Creation

- **Polynomial Features**: x^2, x^3, $x_1 \times x_2$
- **Interaction Features**: Combining features
- **Domain-Specific Features**: Based on expertise

2. Feature Transformation

- **Log Transformation**: For skewed data

- **Box-Cox Transformation**: Normalize distributions
- **Binning**: Continuous to categorical

3. Feature Selection

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

4. Feature Extraction

- **PCA**: Linear combinations
- **LDA**: Maximizes class separation
- **Autoencoders**: Non-linear extraction

Model Evaluation Metrics

Classification Metrics

1. Accuracy

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

2. Precision

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

3. Recall (Sensitivity)

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

4. F1-Score

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

5. ROC-AUC

- Area Under ROC Curve
- Trade-off between TPR and FPR

6. Confusion Matrix

- Visualizes TP, TN, FP, FN

Regression Metrics

1. Mean Absolute Error (MAE)

```
MAE = (1/n) × Σ|y_true - y_pred|
```

2. Mean Squared Error (MSE)

```
MSE = (1/n) × Σ(y_true - y_pred)²
```

3. Root Mean Squared Error (RMSE)

```
RMSE = √MSE
```

4. R^2 Score

```
R² = 1 - (SS_res / SS_tot)
```

Clustering Metrics

- **Silhouette Score**: Cohesion vs separation
- **Davies-Bouldin Index**: Cluster similarity
- **Calinski-Harabasz Index**: Ratio of dispersions

Model Selection & Validation

1. Train-Test Split

- Typical split: 70-30 or 80-20
- Stratified split for imbalanced data

2. Cross-Validation

- **k-Fold**: Data split into k folds
- **Stratified k-Fold**: Maintains class distribution
- **Leave-One-Out (LOO)**: k = n
- **Time Series Split**: For temporal data

3. Hyperparameter Tuning

- **Grid Search**: Exhaustive search
- **Random Search**: Random combinations
- **Bayesian Optimization**: Probabilistic model
- **Genetic Algorithms**: Evolutionary approach

4. Model Selection Criteria

- **Bias-Variance Trade-off**
- **Occam's Razor**: Simpler is better
- **Cross-validation scores**
- **Business requirements**

Common Challenges & Solutions

1. Overfitting

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

- Regularization (L1, L2)
- Dropout (neural networks)
- Early stopping
- More training data
- Simpler models
- Cross-validation

2. Underfitting

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

- More complex models
- Feature engineering
- Reduce regularization

- Increase training time

3. Data Leakage

Prevention:

- Proper train-test split
- Avoid using future information
- Careful feature engineering

4. Curse of Dimensionality

Solutions:

- Feature selection
- Dimensionality reduction
- Regularization

5. Computational Constraints

Solutions:

- Feature selection
- Model simplification
- Distributed computing
- Incremental learning

Deep Learning Overview

Neural Network Basics

- **Perceptron**: Single neuron
- **Multi-Layer Perceptron (MLP)**: Feedforward network
- **Activation Functions**: ReLU, Sigmoid, Tanh

Common Architectures

1. Convolutional Neural Networks (CNN)

- **Use Case**: Image processing
- **Key Components**: Conv layers, pooling, filters

2. Recurrent Neural Networks (RNN)

- **Use Case**: Sequential data

- **Variants**: LSTM, GRU

3. Transformers

- **Use Case**: NLP, sequential data
- **Key Innovation**: Self-attention mechanism

4. Autoencoders

- **Use Case**: Dimensionality reduction, denoising
- **Types**: Vanilla, Variational (VAE)

5. Generative Adversarial Networks (GAN)

- **Use Case**: Data generation
- **Components**: Generator, Discriminator

Deep Learning Best Practices

- **Batch Normalization**: Stabilize training
- **Dropout**: Prevent overfitting
- **Learning Rate Scheduling**: Adaptive learning
- **Transfer Learning**: Use pre-trained models
- **Data Augmentation**: Increase dataset size

Quick Decision Guide

Choosing an Algorithm

For Classification:

- Small dataset → Naive Bayes, SVM
- Interpretability needed → Decision Tree, Logistic Regression
- High accuracy → Ensemble methods (Random Forest, XGBoost)
- Text data → Naive Bayes, SVM, Deep Learning

For Regression:

- Linear relationship → Linear Regression
- Non-linear → Random Forest, SVR, Neural Networks
- Interpretability → Linear Regression, Decision Trees

For Clustering:

- Known k → K-Means
- Unknown k → DBSCAN, Hierarchical
- Large dataset → Mini-batch K-Means

For Dimensionality Reduction:

- Linear → PCA
- Visualization → t-SNE, UMAP
- Feature selection → Lasso, Tree importance

Performance Optimization Checklist

1. ✓ Data quality check
2. ✓ Appropriate preprocessing
3. ✓ Feature engineering
4. ✓ Algorithm selection
5. ✓ Hyperparameter tuning
6. ✓ Cross-validation
7. ✓ Ensemble methods
8. ✓ Error analysis
9. ✓ Business metric alignment
10. ✓ Model monitoring

Resources for Implementation

Python Libraries

- **General ML**: scikit-learn, XGBoost, LightGBM
- **Deep Learning**: TensorFlow, PyTorch, Keras
- **Data Processing**: pandas, NumPy
- **Visualization**: matplotlib, seaborn, plotly
- **AutoML**: auto-sklearn, H2O, TPOT

Key Formulas Reference

- **Bias**: $E[\hat{f}(x)] - f(x)$
- **Variance**: $E[(\hat{f}(x) - E[\hat{f}(x)])^2]$
- **Information Gain**: H(parent) - Σ(weighted H(children))
- **Gini Index**: $1 - \Sigma(p_i)^2$

- **Entropy**: $-\Sigma(p\_i \times \log_2(p\_i))$