**1. Linear Regression (Hồi quy Tuyến tính)**

**1.1. Lý thuyết và Cách vận hành**

Linear Regression là một trong những thuật toán cơ bản nhất trong Machine Learning, được sử dụng để dự đoán giá trị liên tục dựa trên mối quan hệ tuyến tính giữa biến phụ thuộc và các biến độc lập.

* + 1. **Công thức:**
* Đơn biến: y = β₀ + β₁x + ε
* Đa biến: y = β₀ + β₁x₁ + β₂x₂ + ... + βₙxₙ + ε

Trong đó:

* y: biến phụ thuộc (target)
* x: biến độc lập (features)
* β: hệ số hồi quy
* ε: sai số ngẫu nhiên
  + 1. **Cách thức hoạt động:**

1. Thu thập dữ liệu training với các cặp (x,y)
2. Tìm đường thẳng tối ưu bằng phương pháp Least Squares
3. Tối thiểu hóa Sum of Squared Errors (SSE)
4. Sử dụng mô hình để dự đoán giá trị mới
   1. **Ưu điểm và nhược điểm**
      1. **Ưu điểm**

* Đơn giản, dễ hiểu và triển khai
* Tính toán nhanh, ít tốn tài nguyên
* Hoạt động tốt với dữ liệu có mối quan hệ tuyến tính
* Ít bị overfitting với dữ liệu nhỏ
  + 1. **Nhược điểm**
* Chỉ mô hình hóa được mối quan hệ tuyến tính
* Nhạy cảm với outliers
* Giả định về tính độc lập của residuals
* Yêu cầu feature scaling cho đa biến
* Kém hiệu quả với dữ liệu phi tuyến tính
* Có thể bị underfitting với dữ liệu phức tạp
  1. **Thông số đánh giá**
* **R² (Coefficient of Determination):** Đo lường tỷ lệ phương sai được giải thích
* **MSE (Mean Squared Error):** Trung bình bình phương sai số
* **RMSE (Root Mean Squared Error):** Căn bậc hai của MSE
* **MAE (Mean Absolute Error):** Trung bình giá trị tuyệt đối sai số
* **MAPE (Mean Absolute Percentage Error):** Tỷ lệ phần trăm sai số tuyệt đối

**Gợi ý hình ảnh cần tìm**

* Đồ thị scatter plot với đường hồi quy tuyến tính
* Biểu đồ residual plots
* Hình minh họa least squares method
* Đồ thị so sánh predicted vs actual values

**2. Decision Tree (Cây Quyết định)**

**2.1. Lý thuyết**

Decision Tree là một thuật toán supervised learning có thể sử dụng cho cả classification và regression. Nó mô hình hóa quyết định thông qua một cấu trúc cây, trong đó mỗi node trong (internal node) đại diện cho một test trên attribute, mỗi nhánh đại diện cho kết quả của test, và mỗi leaf node đại diện cho class label hoặc giá trị dự đoán.

**2.1.1. Cấu trúc cây:**

* **Root Node:** Node gốc, chứa toàn bộ dataset
* **Internal Nodes:** Các node quyết định, chứa điều kiện test
* **Leaf Nodes:** Các node lá, chứa kết quả cuối cùng
* **Branches:** Các nhánh kết nối nodes

**2.1.2. Quá trình xây dựng cây:**

1. **Chọn Best Attribute:** Sử dụng metrics như Information Gain, Gini Impurity
2. **Split Data:** Chia dataset theo attribute được chọn
3. **Recursive Partitioning:** Lặp lại quá trình cho các subset
4. **Stopping Criteria:** Dừng khi đạt điều kiện (max depth, min samples, purity)
5. **Pruning:** Cắt tỉa để tránh overfitting

**2.1.3. Thuật toán phổ biến:**

* **ID3:** Sử dụng Information Gain
* **C4.5:** Cải tiến ID3, xử lý continuous values và missing data
* **CART:** Classification and Regression Trees, sử dụng Gini Impurity
  1. **Ưu điểm và nhược điểm**
     1. **Ưu điểm**
* **Dễ hiểu và giải thích:** Có thể visualize và interpret dễ dàng
* **Không cần assumptions:** Không giả định về phân phối dữ liệu
* **Xử lý cả numerical và categorical data**
* **Không cần feature scaling**
* **Tự động feature selection**
* **Xử lý được missing values** (với một số implementations)
* **Fast prediction:** O(log n) complexity
* **Non-linear relationships:** Có thể capture non-linear patterns
* **Multi-output support:** Có thể predict multiple outputs simultaneously
  + 1. **Nhược điểm**
* **Overfitting:** Dễ tạo ra cây quá phức tạp
* **Instability:** Nhạy cảm với small changes trong data
* **Bias:** Thiên về attributes có nhiều levels
* **Difficulty with linear relationships:** Kém với linear boundaries
* **Limited expressiveness:** Rectangular decision boundaries only
* **Greedy algorithm limitations:** Local optimal choices
* **Imbalanced data issues:** Bias towards majority class
* **Continuous numerical values:** Cần discretization strategies
  + 1. **Cải thiện thuật toán**
* **Pruning:** Pre-pruning và Post-pruning
* **Ensemble methods:** Random Forest, Boosting
* **Feature engineering:** Transform features cho phù hợp
* **Cross-validation:** Để chọn optimal parameters
* **Cost-sensitive learning:** Cho imbalanced datasets
  1. **Thông số đánh giá**

**Classification:**

* **Accuracy:** Overall correctness
* **Precision/Recall/F1:** Cho từng class
* **Confusion Matrix:** Chi tiết classification results
* **ROC/AUC:** Cho binary classification
* **Tree-specific metrics:**
  + Tree depth
  + Number of nodes/leaves
  + Tree complexity (number of rules)

**Regression:**

* **MSE/RMSE:** Mean squared error metrics
* **MAE:** Mean absolute error
* **R²:** Coefficient of determination
* **Tree-specific:**
  + Variance reduction at each split
  + Node purity measures

**Model Complexity:**

* **Depth:** Số levels trong cây
* **Size:** Tổng số nodes
* **Rules:** Số rules được tạo ra

**Hyperparameters quan trọng**

* **max\_depth:** Độ sâu tối đa của cây
* **min\_samples\_split:** Số samples tối thiểu để split node
* **min\_samples\_leaf:** Số samples tối thiểu tại leaf node
* **max\_features:** Số features xem xét cho best split
* **criterion:** Splitting criterion (gini, entropy, mse)
* **splitter:** Strategy để choose split (best, random)
* **min\_impurity\_decrease:** Minimum impurity decrease để split

**Gợi ý hình ảnh cần tìm**

* Sơ đồ cấu trúc decision tree với nodes và branches
* Visualization của decision boundaries trong 2D space
* Tree traversal flowchart cho prediction process
* Information gain/Gini impurity calculation examples
* Before/after pruning tree comparison
* Decision tree overfitting vs optimal complexity
* Feature importance bar charts
* Tree depth vs accuracy curves

**3. AdaBoost (Adaptive Boosting)**

**3.1. Lý thuyết**

AdaBoost là một thuật toán ensemble learning thuộc nhóm boosting, tập trung vào việc cải thiện hiệu suất bằng cách học từ những sai lầm của các weak learners trước đó.

**3.1.2. Nguyên lý hoạt động:**

1. **Khởi tạo:** Gán trọng số bằng nhau cho tất cả training samples
2. **Training Weak Learner:** Train một classifier đơn giản (thường là decision stump)
3. **Tính Error Rate:** Đánh giá hiệu suất của weak learner
4. **Tính Alpha:** Xác định trọng số cho weak learner trong final model
5. **Update Weights:** Tăng trọng số cho samples bị classify sai
6. **Repeat:** Lặp lại quá trình với weak learner mới
7. **Final Model:** Kết hợp tất cả weak learners với trọng số alpha

**3.1.3. Công thức cập nhật trọng số:**

* α\_t = 0.5 × ln((1 - ε\_t) / ε\_t)
* w\_i^(t+1) = w\_i^(t) × exp(-α\_t × y\_i × h\_t(x\_i))

**3.2 Ưu điểm và nhược điểm**

**3.2.1. Ưu điểm**

* Hiệu suất cao với weak learners đơn giản
* Giảm bias và variance
* Không cần feature selection phức tạp
* Tự động focus vào hard examples
* Ít tham số cần tuning
* Hoạt động tốt với dữ liệu không cân bằng
* Fast training với weak learners đơn giản

**3.2.1. Nhược điểm**

* Rất nhạy cảm với noise và outliers
* Dễ bị overfitting với dữ liệu noisy
* Hiệu suất phụ thuộc vào weak learner base
* Có thể unstable với dữ liệu nhỏ
* Khó tune hyperparameters
* Không song song hóa được (sequential nature)
* Kém hiệu quả với dữ liệu có nhiều irrelevant features

**3.3. Thông số đánh giá**

**Classification:**

* Training Error: Giảm theo từng iteration
* Test Error: Monitor để tránh overfitting
* Weighted Error Rate: Sai số có trọng số của từng weak learner
* Accuracy, Precision, Recall, F1-Score
* Learning Curves: Theo dõi hiệu suất qua iterations

**Monitoring:**

* Alpha values: Trọng số của từng weak learner
* Sample weights distribution: Phân bố trọng số samples
* Convergence plots: Đồ thị hội tụ của algorithm

**Gợi ý hình ảnh cần tìm**

* Sơ đồ minh họa AdaBoost algorithm workflow
* Biểu đồ trọng số samples qua các iterations
* Learning curves showing error reduction
* Hình ảnh decision boundary evolution
* Comparison của weak learners vs final strong learner

**4. XGBoost (eXtreme Gradient Boosting)**

**4.1. Lý thuyết và Cách vận hành**

XGBoost là một implementation tối ưu của Gradient Boosting, được thiết kế để đạt hiệu suất cao và tốc độ nhanh. Nó sử dụng gradient descent để tối ưu hóa loss function.

**4.1.1. Nguyên lý hoạt động:**

1. **Khởi tạo:** Bắt đầu với prediction đơn giản (thường là mean)
2. **Tính Gradients:** Tính first và second order gradients của loss function
3. **Build Tree:** Xây dựng tree để fit residuals/gradients
4. **Regularization:** Áp dụng L1/L2 regularization để tránh overfitting
5. **Update Model:** Thêm tree mới vào ensemble với learning rate
6. **Repeat:** Lặp lại cho đến khi đạt stopping criteria

**4.1.2. Key Features:**

* **Regularization:** L1 (Lasso) và L2 (Ridge) regularization
* **Pruning:** Pre-pruning và post-pruning
* **Parallel Processing:** Tận dụng multi-threading
* **Cross-validation:** Built-in CV support
* **Missing Value Handling:** Tự động xử lý missing values

**4.1.3. Tham số quan trọng:**

* n\_estimators: số boosting rounds
* learning\_rate: shrinkage để tránh overfitting
* max\_depth: độ sâu tối đa của trees
* subsample: tỷ lệ samples cho mỗi tree
* colsample\_bytree: tỷ lệ features cho mỗi tree
* reg\_alpha, reg\_lambda: L1, L2 regularization

**4.2. Ưu điểm và nhược điểm**

**4.2.1. Ưu điểm**

* Hiệu suất rất cao, thường top trong competitions
* Tốc độ training nhanh nhất trong các gradient boosting
* Built-in regularization giảm overfitting
* Xử lý missing values tự động
* Support cả classification và regression
* Feature importance và interpretability tools
* Memory efficient
* Cross-platform compatibility
* Hỗ trợ GPU acceleration

**4.2.2. Nhược điểm**

* Nhiều hyperparameters cần tuning
* Có thể overfit nếu không tune cẩn thận
* Nhạy cảm với outliers
* Yêu cầu feature engineering tốt
* Black box model (khó giải thích)
* Memory intensive với datasets lớn
* Có thể bias với imbalanced data
* Cần experience để tune hiệu quả

**4.3. Thông số đánh giá**

**Classification:**

* **Logloss:** Logarithmic loss cho probabilistic predictions
* **AUC:** Area Under Curve
* **Accuracy, Precision, Recall, F1-Score**
* **Multi-class Logloss:** Cho multi-class problems

**Regression:**

* **RMSE:** Root Mean Squared Error
* **MAE:** Mean Absolute Error
* **MAPE:** Mean Absolute Percentage Error

**XGBoost Specific:**

* **Training History:** Loss theo từng iteration
* **Feature Importance:** Gain, Cover, Frequency
* **SHAP Values:** Cho model interpretability
* **Learning Curves:** Validation vs training error

**Hyperparameter Tuning Strategy**

1. **Cố định:** n\_estimators = 1000, learning\_rate = 0.1
2. **Tune tree parameters:** max\_depth, min\_child\_weight
3. **Tune regularization:** reg\_alpha, reg\_lambda
4. **Tune sampling:** subsample, colsample\_bytree
5. **Lower learning\_rate và tăng n\_estimators**

**Gợi ý hình ảnh cần tìm**

* XGBoost architecture diagram với boosting process
* Feature importance plots (gain, cover, frequency)
* Learning curves cho training vs validation
* Hyperparameter tuning heatmaps
* SHAP summary plots cho model interpretation
* Performance comparison với other algorithms
* GPU vs CPU training speed comparison