# BÁO CÁO LÝ THUYẾT – CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY

## 1️⃣ Quy trình khai phá dữ liệu CRISP–DM và SEMMA

🔹 CRISP–DM (Cross Industry Standard Process for Data Mining): Là mô hình quy trình chuẩn công nghiệp cho khai phá dữ liệu, gồm 6 giai đoạn:

|  |  |  |
| --- | --- | --- |
| Bước | Tên giai đoạn | Nội dung |
| 1 | Business Understanding | Hiểu rõ mục tiêu nghiệp vụ và xác định vấn đề cần giải quyết. |
| 2 | Data Understanding | Thu thập dữ liệu, kiểm tra chất lượng, khám phá dữ liệu. |
| 3 | Data Preparation | Làm sạch, xử lý dữ liệu, chọn đặc trưng phù hợp cho mô hình. |
| 4 | Modeling | Áp dụng thuật toán (Cây quyết định, Hồi quy, KNN, v.v.) để xây dựng mô hình. |
| 5 | Evaluation | Đánh giá kết quả, đảm bảo mô hình đạt mục tiêu nghiệp vụ. |
| 6 | Deployment | Triển khai mô hình trong thực tế hoặc tạo báo cáo. |

SEMMA (Sample, Explore, Modify, Model, Assess): Là quy trình của SAS Institute, tập trung vào phân tích dữ liệu. Gồm 5 bước:

|  |  |  |
| --- | --- | --- |
| Bước | Ý nghĩa | Mục tiêu |
| Sample | Lấy mẫu dữ liệu đại diện | Giảm thời gian xử lý |
| Explore | Khám phá dữ liệu | Phân tích thống kê, tìm mẫu |
| Modify | Biến đổi dữ liệu | Làm sạch, chọn đặc trưng |
| Model | Xây dựng mô hình | Áp dụng thuật toán học máy |
| Assess | Đánh giá mô hình | Chọn mô hình tốt nhất |

## 2️⃣ Cây quyết định hoạt động như thế nào?

Cây quyết định (Decision Tree) là mô hình học máy dạng cây, sử dụng để phân loại hoặc hồi quy.

• Nút gốc (Root node): Nút đầu tiên chứa toàn bộ dữ liệu.

• Nút trong (Internal node): Thể hiện điều kiện phân tách, ví dụ: age > 30.

• Nhánh (Branch): Kết quả của điều kiện (Đúng / Sai).

• Nút lá (Leaf node): Kết quả dự đoán cuối cùng.

## 3️⃣ Các tiêu chí phân tách (Gini, Entropy, Information Gain)

Các tiêu chí giúp chọn thuộc tính tốt nhất để chia dữ liệu tại mỗi nút:

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | Ý nghĩa | Công thức / Trực giác | Khi nào tốt |
| Gini Index | Đo độ tinh khiết của nút | G = 1 - Σ p\_i² | Nhỏ càng tốt |
| Entropy | Đo độ hỗn loạn dữ liệu | H = -Σ p\_i log₂(p\_i) | Nhỏ càng tốt |
| Information Gain | Lượng thông tin thu được sau chia | IG = H\_parent - Σ w\_i H\_i | Lớn càng tốt |

## 4️⃣ Rừng cây (Random Forest) là gì?

Random Forest là tập hợp nhiều cây quyết định độc lập, mỗi cây được huấn luyện trên mẫu dữ liệu ngẫu nhiên. Kết quả cuối cùng là trung bình (hồi quy) hoặc bỏ phiếu (phân loại).

## 5️⃣ Ưu điểm và hạn chế

|  |  |  |
| --- | --- | --- |
| Mô hình | Ưu điểm | Hạn chế |
| Cây quyết định | Dễ hiểu, trực quan, không cần chuẩn hóa dữ liệu. | Dễ overfit, nhạy với dữ liệu nhiễu. |
| Random Forest | Hiệu suất cao, chống overfit, ổn định. | Tốn tài nguyên, khó giải thích. |

## 6️⃣ Code mẫu mô hình Cây quyết định (Python, Scikit-learn)

from sklearn.tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
X = df.drop('target', axis=1)  
y = df['target']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = DecisionTreeClassifier(max\_depth=4, criterion='gini', random\_state=42)  
model.fit(X\_train, y\_train)  
  
y\_pred = model.predict(X\_test)  
print("Độ chính xác:", accuracy\_score(y\_test, y\_pred))

## 7️⃣ Triển khai mô hình Random Forest trong Python

from sklearn.ensemble import RandomForestClassifier  
  
rf = RandomForestClassifier(  
 n\_estimators=100,  
 max\_depth=None,  
 random\_state=42  
)  
  
rf.fit(X\_train, y\_train)  
y\_pred = rf.predict(X\_test)  
print("Độ chính xác:", accuracy\_score(y\_test, y\_pred))

## 8️⃣ Đánh giá tầm quan trọng của các đặc trưng

import pandas as pd  
import matplotlib.pyplot as plt  
  
feat\_imp = pd.Series(rf.feature\_importances\_, index=X.columns)  
feat\_imp.sort\_values(ascending=True).plot.barh(figsize=(6,4))  
plt.title("Tầm quan trọng của các đặc trưng (Random Forest)")  
plt.show()

## 9️⃣ Điều chỉnh siêu tham số bằng GridSearchCV

from sklearn.model\_selection import GridSearchCV  
  
param\_grid = {  
 'n\_estimators': [50, 100, 150],  
 'max\_depth': [3, 5, 8, None]  
}  
  
grid = GridSearchCV(  
 estimator=RandomForestClassifier(random\_state=42),  
 param\_grid=param\_grid,  
 scoring='roc\_auc',  
 cv=5  
)  
  
grid.fit(X\_train, y\_train)  
print("Tham số tối ưu:", grid.best\_params\_)  
print("AUC tốt nhất:", grid.best\_score\_)