1. Quy trình khai phá dữ liệu CRISP – DM (Cross Industry Standard Process for Data Mining) là gì ? Quy trình khai phá dữ liệu SEMMA (Sample, Explore, Modify, Model, Access) là gì?

- Quy trình khai phá dữ liệu CRISP-DM là một mô hình quy trình toàn diện, linh hoạt, và được sử dụng rộng rãi nhất. Nó chia dự án khai phá dữ liệu thành sáu giai đoạn chính theo trình tự tuần tự nhưng có thể lặp lại (lặp đi lặp lại).

Giai đoạn	Mục tiêu chính	Mô tả
1. Hiểu biết về Kinh doanh	Xác định mục tiêu và yêu cầu dự án.	Tập trung vào việc hiểu rõ vấn đề kinh doanh và chuyển đổi nó thành các mục tiêu của dự án khai phá dữ liệu.
2. Hiểu biết về Dữ liệu	Thu thập, khám phá và đánh giá chất lượng dữ liệu.	Kiểm tra nguồn dữ liệu, xác định các vấn đề về chất lượng (thiếu, ngoại lai), và làm quen với dữ liệu.
3. Chuẩn bị Dữ liệu	Xây dựng tập dữ liệu cuối cùng cho mô hình.	Bao gồm các bước làm sạch, chuyển đổi, tích hợp dữ liệu và xử lý các giá trị thiếu hoặc ngoại lai. Đây thường là giai đoạn tốn nhiều thời gian nhất.
4. Xây dựng Mô hình	Lựa chọn và áp dụng các kỹ thuật khai phá dữ liệu.	Lựa chọn thuật toán (hồi quy, phân loại, gom cụm...), hiệu chỉnh các tham số, và đánh giá sơ bộ mô hình.
5. Đánh giá Mô hình	Đánh giá kỹ lưỡng mô hình và kết quả dựa trên mục tiêu kinh doanh.	Đảm bảo mô hình đáp ứng các tiêu chí về chất lượng (độ chính xác, độ tin cậy) và giải quyết được vấn đề kinh doanh đã đặt ra.
6. Triển khai	Đưa kết quả vào sử dụng thực tế.	Kết quả mô hình có thể được triển khai dưới dạng báo cáo, công cụ phần mềm, hoặc tích hợp vào quy trình kinh doanh hiện tại.

- SEMMA là một khung phương pháp luận được phát triển bởi SAS Institute (một công ty phần mềm thống kê). Nó tập trung nhiều hơn vào các bước kỹ thuật cần thiết để xây dựng mô hình khai phá dữ liệu từ một tập dữ liệu.

Giai đoạn	Mục tiêu chính	Chi tiết kĩ thuật

Sample	Chọn một tập dữ liệu con (mẫu) đại diện.	Lựa chọn một tập hợp con dữ liệu đủ lớn để phân tích nhưng đủ nhỏ để xử lý hiệu quả.
Explore	Tìm hiểu dữ liệu thông qua thống kê mô tả và trực quan hóa.	Phân tích phân phối, mối quan hệ giữa các biến, và xác định các bất thường hoặc giá trị ngoại lai.
Modify	Tiền xử lý dữ liệu và tạo biến mới.	Thực hiện các kỹ thuật tiền xử lý (xử lý ngoại lai, giá trị thiếu) và tạo ra các đặc trưng/biến (feature engineering) mới để cải thiện mô hình.
Model	Áp dụng các kỹ thuật mô hình hóa.	Xây dựng các mô hình dự đoán hoặc mô tả (ví dụ: hồi quy, cây quyết định, mạng nơ-ron).
Access	Đánh giá hiệu suất và độ tin cậy của mô hình.	Đánh giá tính hữu dụng của mô hình đối với mục tiêu kinh doanh và đo lường hiệu suất mô hình trên dữ liệu mới.

2. Cây quyết định hoạt động như thế nào? Hãy giải thích các thành phần chính (nút gốc, nút lá, nhánh) và cách cây đưa ra dự đoán.

- Cây Quyết Định (Decision Tree) là một thuật toán học máy có giám sát (Supervised Learning) mạnh mẽ và trực quan, được sử dụng cho cả bài toán phân loại (Classification) và hồi quy (Regression). Nó hoạt động bằng cách chia tập dữ liệu thành các tập con nhỏ hơn dựa trên các quy tắc quyết định đơn giản, được xây dựng theo cấu trúc giống như một cây.
- Gồm 3 loại nút và các nhánh:
    + Nút gốc

Vị trí: Nút đầu tiên, nằm trên cùng của cây

Chức năng: Đại diện cho toàn bộ tập dữ liệu ban đầu. Đây là điểm bắt đầu của quá trình quyết định và chứa **thuộc tính (feature) tốt nhất** để phân chia dữ liệu.

    + Nhánh

Vị trí: Các đường nối giữa các nút.

Chức năng: Đại diện cho kết quả của một bài kiểm tra hoặc một quy tắc quyết định. Ví dụ: nếu nút gốc là "Độ tuổi", các nhánh có thể là "Độ tuổi < 30" và "Độ tuổi ≥ 30".

    + Nút lá

Vị trí: Các nút nằm ở cuối cùng của cây.

Chức năng: Đại diện cho **kết quả cuối cùng** hoặc **nhãn phân loại (class label)**. Khi thuật toán đạt đến nút lá, quá trình ra quyết định kết thúc và dự đoán được đưa ra.

- Quá trình Hoạt động: Xây dựng và Đưa ra Dự đoán

1. Cách Cây được Xây dựng (Training)

Quá trình xây dựng cây được thực hiện lặp đi lặp lại (recursive partitioning) thông qua các bước sau:

- **Lựa chọn Thuộc tính Tốt nhất:** Tại mỗi nút (bao gồm cả nút gốc), thuật toán phải chọn một thuộc tính để chia dữ liệu sao cho các tập con tạo ra là **thuần nhất** (homogeneous) nhất có thể về nhãn phân loại.
- **Tiêu chí Phân tách (Splitting Criteria):** Để đo lường độ thuần khiết, các thuật toán sử dụng các chỉ số như:
    - **Entropy** và **Information Gain** (Độ lợi Thông tin): Được sử dụng để tìm kiếm sự giảm thiểu độ hỗn loạn sau khi phân tách.
    - **Chỉ số Gini (Gini Index/Gini Impurity):** Đo lường xác suất một phần tử được chọn ngẫu nhiên bị gán sai nhãn nếu nó được gán nhãn theo phân phối của tập con.
- **Ngừng Phân tách:** Quá trình phân tách tiếp diễn cho đến khi một trong các điều kiện dừng được đáp ứng:
    - Tất cả các quan sát trong một nút thuộc cùng một lớp (nút thuần nhất).
    - Số lượng quan sát trong một nút đạt đến một ngưỡng tối thiểu (Minimal Node Size).
    - Độ sâu tối đa của cây đã đạt được (Max Depth).

2. Cách Cây đưa ra Dự đoán (Prediction)

Khi một điểm dữ liệu mới (chưa được gán nhãn) được đưa vào cây, quá trình dự đoán diễn ra như sau:

1. **Bắt đầu từ Nút Gốc:** Điểm dữ liệu được kiểm tra theo quy tắc tại Nút Gốc.
2. **Đi theo Nhánh:** Dựa trên kết quả của bài kiểm tra, điểm dữ liệu sẽ di chuyển xuống nhánh tương ứng.
3. **Lặp lại:** Quá trình kiểm tra và di chuyển qua các nhánh lặp lại tại mỗi Nút Bên trong tiếp theo.
4. **Đạt Nút Lá:** Khi điểm dữ liệu đến một Nút Lá, giá trị của Nút Lá đó được gán làm dự đoán cuối cùng:
    - **Phân loại:** Dự đoán là **lớp (class)** chiếm đa số trong Nút Lá đó.
    - **Hồi quy:** Dự đoán là **giá trị trung bình** của tất cả các giá trị mục tiêu trong Nút Lá đó.

3. Các tiêu chí phân tách (splitting criteria) như Gini Index, Entropy, hay Information Gain được sử dụng trong cây quyết định là gì? Chúng khác nhau ra sao?

- Các tiêu chí phân tách như **Gini Index**, **Entropy**, và **Information Gain** là các thước đo toán học được sử dụng trong thuật toán Cây Quyết Định (**Decision Tree**) để xác định **thuộc tính tốt nhất** để chia (split) dữ liệu tại mỗi nút. Mục tiêu chung của chúng là tìm cách phân tách dữ liệu sao cho các tập con (nút con) được tạo ra trở nên **thuần nhất (pure)** nhất có thể về nhãn lớp (class label).

Đặc điểm	Gini Index	Entropy
Phép tính	Sử dụng phép toán bình phương	Sử dụng phép toán Logarit
Tốc độ tính toán	Thường nhanh hơn vì không liên quan đến logarit, vốn tốn thời gian tính toán hơn.	Chậm hơn do phải tính toán logarit.
Ứng dụng	Là tiêu chí mặc định được sử dụng trong thuật toán CART (Classification and Regression Trees).	Là tiêu chí chính được sử dụng trong thuật toán ID3 và C4.5.
Xu hướng	Có xu hướng ưu tiên các phân tách tạo ra các nhóm có kích thước bằng nhau hơn (mặc dù không phải luôn luôn).	Có xu hướng ưu tiên các thuộc tính có nhiều giá trị (nhiều nút con) hơn, dẫn đến nguy cơ overfitting (quá khớp).

4. Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?

Rừng cây (Random Forest) là một mô hình học máy dạng tập hợp (ensemble) gồm nhiều **cây quyết định (Decision Trees)**. Mỗi cây được huấn luyện trên một **mẫu dữ liệu ngẫu nhiên** và **tập con ngẫu nhiên các đặc trưng**, sau đó các cây cùng **bỏ phiếu** để đưa ra kết quả cuối cùng (phân loại) hoặc **lấy trung bình** (hồi quy).
So với **một cây quyết định đơn lẻ**, Random Forest:

- **Ổn định và chính xác hơn**, vì trung bình hóa nhiều mô hình khác nhau.
- **Giảm hiện tượng quá khớp (overfitting)** nhờ tính ngẫu nhiên khi huấn luyện.
- **Khả năng tổng quát hóa tốt hơn** khi áp dụng cho dữ liệu mới.
  Cung cấp thông tin về tầm quan trọng của đặc trưng (feature importance).

Nhờ các ưu điểm trên, Random Forest thường cho **hiệu suất cao và kết quả dự báo đáng tin cậy hơn** so với cây quyết định đơn lẻ trong các bài toán phân loại và hồi quy.

5. Những ưu điểm và hạn chế của cây quyết định và Random Forest là gì? Trong trường hợp nào thì cây quyết định có thể hoạt động kém hiệu quả?

- Ưu điểm và hạn chế của cây quyết định:

Ưu điểm	Hạn chế
Dễ diễn giải	Overfitting

Cấu trúc giống như sơ đồ tư duy (flowchart) giúp dễ hiểu quá trình ra quyết định.	Rất nhạy cảm với dữ liệu huấn luyện. Một thay đổi nhỏ có thể dẫn đến một cấu trúc cây hoàn toàn khác, dễ bị quá khớp với dữ liệu huấn luyện và hoạt động kém trên dữ liệu mới.
Yêu cầu dữ liệu tối thiểu	Tính biến động cao (High Variance)
Không yêu cầu chuẩn hóa hoặc co giãn (scaling) dữ liệu. Có thể xử lý cả dữ liệu định tính và định lượng.	Tính toán dự đoán có thể không ổn định; dễ thay đổi nếu thêm/bớt dữ liệu huấn luyện.
Chi phí tính toán thấp	Thiên vị với dữ liệu lệch
Nhanh chóng để huấn luyện và đưa ra dự đoán.	Các tiêu chí phân tách (như Information Gain) ưu tiên các thuộc tính có nhiều giá trị hoặc các lớp chiếm đa số, dẫn đến cây bị lệch.

- Ưu điểm và hạn chế của Rừng ngẫu nhiên:

Ưu điểm	Hạn chế
Giảm quá khớp	Tốc độ chậm hơn
Nhờ kết hợp nhiều cây và lấy trung bình/bỏ phiếu, nó làm giảm đáng kể phương sai (variance) so với một cây đơn lẻ, dẫn đến mô hình tổng quát hóa tốt hơn.	Do phải huấn luyện nhiều cây quyết định, thời gian huấn luyện và đưa ra dự đoán sẽ lâu hơn đáng kể so với một cây đơn lẻ.
Độ chính xác cao	Kém diễn giải hơn
Thường là một trong những thuật toán học máy có hiệu suất dự đoán tốt nhất.	Là một "hộp đen" (black box). Việc diễn giải các quy tắc của hàng trăm cây để hiểu quá trình ra quyết định là rất khó.
Xử lý hiệu quả dữ liệu thiếu và ngoại lai	Tiêu tốn bộ nhớ
Hiệu suất của nó thường không bị ảnh hưởng nhiều bởi việc xử lý dữ liệu thiếu hoặc các giá trị ngoại lai.	Việc lưu trữ hàng trăm cây yêu cầu bộ nhớ (RAM) đáng kể.

- Trường hợp Cây Quyết Định Hoạt động Kém Hiệu quả:
    + **Mối quan hệ phức tạp và phi tuyến tính:** Khi dữ liệu có mối quan hệ phức tạp mà ranh giới quyết định không thể được mô tả bằng các phân tách đơn giản (ví dụ: các đường chéo), cây đơn lẻ có thể cần quá nhiều nút phân tách (sâu) để khớp dữ liệu, dẫn đến quá khớp.
    + **Dữ liệu thiếu cân bằng (Imbalanced Data):** Cây quyết định có xu hướng bị thiên vị về phía lớp chiếm đa số (major class), dẫn đến hiệu suất phân loại kém đối với lớp thiểu số (minority class) quan trọng.
    + **Dữ liệu nhiễu (Noisy Data):** Cây quyết định rất nhạy cảm với nhiễu và các giá trị ngoại lai. Một vài điểm dữ liệu bất thường có thể buộc cây tạo ra một

phân tách hoàn toàn không cần thiết, làm phức tạp cấu trúc cây và gây quá khớp.

+ **Sự biến động dữ liệu cao (High Variance):** Trong các tình huống mà dữ liệu huấn luyện thay đổi nhỏ có thể dẫn đến sự thay đổi lớn trong cấu trúc cây, Cây Quyết Định sẽ đưa ra dự đoán không ổn định trên dữ liệu mới. Đây là lý do chính mà Random Forest (sử dụng Bagging để giảm Variance) lại được ưu tiên hơn.

6. Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình cây quyết định không? Hãy mô tả các bước thực hiện + Làm thế nào để triển khai một mô hình Random Forest trong Python? Bạn thường thiết lập các tham số nào (ví dụ: n_estimators, max_depth)?

```
# Import thư viện
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Ví dụ dữ liệu
df = pd.read_csv("diabetes_prediction_dataset.csv")
# Chuẩn bị dữ liệu
X = df.drop("diabetes", axis=1)
y = df["diabetes"]
# Chia dữ liệu train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Mô hình Cây quyết định
dt = DecisionTreeClassifier(max_depth=4, random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
# Mô hình Rừng cây ngẫu nhiên
rf = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

7. Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong Random Forest bằng Python? +

```
# Lấy tầm quan trọng của đặc trưng
importances = rf.feature_importances_
feature_names = X.columns
# Tạo DataFrame sắp xếp theo mức độ quan trọng
feat_imp = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feat_imp = feat_imp.sort_values('Importance', ascending=False)
print(feat_imp)
# Vẽ biểu đồ trực quan
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
```

```
plt.barh(feat_imp['Feature'], feat_imp['Importance'], color='skyblue')
plt.gca().invert_yaxis()
plt.title("Feature Importance in Random Forest")
plt.xlabel("Importance")
plt.show()
```

8. Điều chỉnh siêu tham số (hyperparameter tuning) cho cây quyết định hoặc Random Forest chưa? Hãy mô tả cách bạn sử dụng GridSearchCV hoặc RandomizedSearchCV

```
from sklearn.model_selection import GridSearchCV
# Định nghĩa các giá trị thử nghiệm
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10]
}
# Gọi mô hình và GridSearchCV
grid = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    scoring='accuracy',
    cv=5,
    verbose=1,
    n_jobs=-1
)
# Huấn luyện
grid.fit(X_train, y_train)
# Kết quả
print("Best Parameters:", grid.best_params_)
print("Best Accuracy:", grid.best_score_)
```