Chương 2

* Hiểu lý do vì sao Python lại được dùng rộng rãi trong học máy: **ngôn ngữ mạnh, dễ dùng, có thể dùng từ nghiên cứu đến sản xuất.**
* Biết các thư viện quan trọng cần học:
  + NumPy, Pandas, Matplotlib để tiền xử lý, khám phá dữ liệu.
  + scikit-learn để xây dựng mô hình học máy.
* Biết cách cài đặt môi trường học máy với Python bằng nhiều cách khác nhau, đặc biệt là **dùng Anaconda để đơn giản hóa quá trình cài đặt.**

**Chương 6**

* Biết cách vẽ và đọc hiểu:
  + **Histogram** để xem phân bố dữ liệu (nhanh và dễ)
  + **Density Plot** để xem phân bố mượt hơn
  + **Boxplot** để xem trung vị, khoảng tứ phân vị và phát hiện ngoại lệ
  + **Correlation matrix** để phát hiện các cặp biến có tương quan cao
  + **Scatter plot matrix** để thấy mối quan hệ trực quan giữa các biến
* Hiểu rằng việc **trực quan hóa dữ liệu là bước quan trọng và bắt buộc** trong bất kỳ dự án học máy nào.
* Biết cách sử dụng matplotlib, pandas, scatter\_matrix để thực hiện trực quan hóa.

**Chương 10**

*  Hiểu rằng việc chọn chỉ số phù hợp là rất quan trọng vì nó ảnh hưởng đến cách đánh giá và lựa chọn thuật toán. Ví dụ, độ chính xác không phù hợp cho các tập dữ liệu mất cân bằng.

**Các chỉ số cụ thể và ứng dụng**:

* Nắm rõ cách sử dụng và ý nghĩa của các chỉ số như accuracy, logloss, AUC, confusion matrix, classification report cho bài toán phân loại, và MAE, MSE, R² cho bài toán hồi quy.
* Học được rằng một số chỉ số (như MSE, logloss) được báo cáo dưới dạng số âm trong scikit-learn, cần chú ý khi diễn giải.

**Thực hành với scikit-learn**:

* Làm quen với cách sử dụng thư viện scikit-learn để tải dữ liệu, chia dữ liệu, huấn luyện mô hình, và đánh giá hiệu suất bằng các chỉ số khác nhau.
* Hiểu cách triển khai **cross-validation** để đánh giá mô hình một cách đáng tin cậy.

**Phân biệt phân loại và hồi quy**:

* Hiểu sự khác biệt giữa các bài toán phân loại (dự đoán lớp) và hồi quy (dự đoán giá trị số), cũng như các chỉ số phù hợp cho từng loại.

**Kỹ năng đọc và diễn giải kết quả**:

* Biết cách diễn giải các kết quả như ma trận nhầm lẫn, báo cáo phân loại, hoặc giá trị R² để đánh giá chất lượng mô hình.

Chương 14

**Hiểu về Pipelines trong scikit-learn**:

* Pipelines là công cụ mạnh mẽ để tự động hóa quy trình máy học, giúp tổ chức các bước từ tiền xử lý đến mô hình hóa một cách rõ ràng và hiệu quả.
* Chúng cho phép tích hợp nhiều bước (transformers và estimators) vào một chuỗi tuyến tính, được xử lý như một mô hình duy nhất.

**Tránh rò rỉ dữ liệu**:

* Hiểu rằng việc áp dụng tiền xử lý (như chuẩn hóa hoặc trích xuất đặc trưng) trên toàn bộ tập dữ liệu trước khi chia thành tập huấn luyện và kiểm tra có thể gây rò rỉ dữ liệu.
* Pipelines đảm bảo rằng các bước tiền xử lý chỉ được thực hiện trên dữ liệu huấn luyện trong mỗi lần đánh giá, giúp đánh giá mô hình chính xác hơn.

**Sử dụng FeatureUnion**:

* FeatureUnion là một công cụ hữu ích để kết hợp nhiều phương pháp trích xuất đặc trưng (như PCA và SelectKBest) thành một tập hợp đặc trưng lớn hơn, sau đó sử dụng để huấn luyện mô hình.
* FeatureUnion cũng được tích hợp trong Pipeline, đảm bảo rằng việc trích xuất đặc trưng chỉ xảy ra trên tập huấn luyện.

**Thực hành với scikit-learn**:

* Làm quen với cách xây dựng Pipeline bằng cách sử dụng danh sách các bộ biến đổi (transformers) và mô hình (estimators).
* Hiểu cách sử dụng cross\_val\_score để đánh giá toàn bộ Pipeline, tương tự như đánh giá một mô hình đơn lẻ.
* Nắm cách kết hợp các công cụ như StandardScaler, PCA, SelectKBest, và FeatureUnion trong một quy trình thống nhất.

**Tư duy tổ chức mã**:

* Pipelines khuyến khích viết mã sạch hơn, dễ bảo trì và tái sử dụng. Tôi học được cách tổ chức các bước xử lý dữ liệu và mô hình hóa một cách logic.
* Có thể mở rộng ý tưởng để tạo các Pipeline phức tạp hơn, ví dụ: lồng Pipelines trong Pipelines.

**Chương 17  
 Tầm quan trọng của việc lưu và tải mô hình:**

* Hiểu rằng lưu mô hình là bước thiết yếu để triển khai hoặc tái sử dụng mô hình trong thực tế, giúp tiết kiệm thời gian huấn luyện.
* Việc này đặc biệt hữu ích khi mô hình đã được tinh chỉnh và sẵn sàng cho sản xuất.

**Sử dụng Pickle và Joblib:**

* Nắm cách sử dụng pickle để serialize và deserialize các mô hình máy học một cách đơn giản.
* Hiểu rằng joblib là lựa chọn tốt hơn khi làm việc với các mô hình có dữ liệu lớn (như mảng NumPy), vì nó hiệu quả hơn trong việc xử lý các cấu trúc dữ liệu này.

**Cách thực hiện trong scikit-learn:**

* Học cách lưu mô hình bằng pickle.dump hoặc joblib.dump và tải mô hình bằng pickle.load hoặc joblib.load.
* Hiểu rằng sau khi tải, mô hình có thể được sử dụng ngay để dự đoán hoặc đánh giá mà không cần huấn luyện lại.

**Lưu ý về tính tương thích:**

* Ghi nhận tầm quan trọng của việc theo dõi phiên bản Python và các thư viện để tránh lỗi khi tải mô hình trên môi trường khác.
* Hiểu rằng có thể cần serialize thủ công các tham số mô hình nếu muốn triển khai trên nền tảng không dùng Python.

**Ứng dụng thực tế:**

* Nhận ra rằng việc lưu và tải mô hình là bước quan trọng trong quy trình triển khai máy học, như tích hợp mô hình vào ứng dụng web hoặc hệ thống sản xuất.

**Chương 5: Understand Your Data With Descriptive Statistics**

**Tầm quan trọng của việc khám phá dữ liệu:**

* Hiểu rằng việc phân tích dữ liệu ban đầu là bước quan trọng để nắm bắt cấu trúc, đặc điểm và các vấn đề tiềm ẩn của tập dữ liệu trước khi áp dụng các thuật toán học máy.
* Giúp phát hiện các giá trị bất thường, dữ liệu bị thiếu, hoặc các mẫu tiềm năng trong dữ liệu, từ đó đưa ra quyết định xử lý phù hợp.

**Các kỹ thuật phân tích dữ liệu:**

* Xem trước dữ liệu: Sử dụng data.head(n) để xem n dòng đầu tiên, giúp hình dung các thuộc tính và giá trị (ví dụ: xem 20 dòng của tập diabetes.csv).
* Kiểm tra kích thước: Sử dụng data.shape để xác định số dòng và số cột (ví dụ: 768 dòng, 9 cột).
* Kiểm tra kiểu dữ liệu: Sử dụng data.dtypes để biết kiểu dữ liệu của từng cột (ví dụ: int64, float64).
* Thống kê mô tả: Sử dụng data.describe() để tính các chỉ số như trung bình, độ lệch chuẩn, phân vị, giá trị min/max, giúp hiểu phân phối và đặc tính của dữ liệu.
* Phân phối lớp: Sử dụng data.groupby().size() để kiểm tra phân phối của các giá trị trong một cột (ví dụ: số lần mang thai trong cột Pregnancies).
* Tương quan thuộc tính: Sử dụng data.corr(method='pearson') để tính ma trận tương quan, xác định mức độ liên quan tuyến tính giữa các cột.
* Độ lệch phân phối: Sử dụng data.skew() để kiểm tra độ lệch của phân phối đơn biến, giúp nhận biết dữ liệu có phân phối lệch trái/phải.

**Cách thực hiện trong pandas:**

* Sử dụng thư viện pandas để đọc dữ liệu từ file CSV (read\_csv) và thực hiện các phân tích trên.
* Điều chỉnh định dạng hiển thị bằng set\_option (ví dụ: display.width, display.precision) để kết quả dễ đọc hơn.
* Các kết quả như thống kê mô tả, ma trận tương quan được in ra để hỗ trợ phân tích.

**Lưu ý về việc phân tích dữ liệu:**

* Cần kiểm tra dữ liệu bị thiếu hoặc bất thường (ví dụ: giá trị 0 trong BloodPressure hoặc BMI có thể không hợp lý).
* Việc hiểu dữ liệu giúp lựa chọn các bước tiền xử lý phù hợp (chuẩn hóa, xử lý giá trị thiếu, v.v.) trước khi huấn luyện mô hình.

**Chương 9: Evaluate the Performance of Machine Learning Algorithms with Resampling**

**Tầm quan trọng của việc đánh giá mô hình:**

* Hiểu rằng việc đánh giá hiệu suất mô hình là cần thiết để đảm bảo mô hình hoạt động tốt trên dữ liệu chưa thấy, tránh hiện tượng quá khớp (overfitting).
* Các kỹ thuật lấy mẫu lại giúp cung cấp ước lượng ổn định và đáng tin cậy về hiệu suất mô hình.

**Các phương pháp lấy mẫu lại:**

* Chia train-test: Sử dụng train\_test\_split để chia dữ liệu thành tập huấn luyện và kiểm tra (ví dụ: 67% train, 33% test), đơn giản nhưng có thể thiếu ổn định.
* K-fold Cross Validation: Sử dụng KFold để chia dữ liệu thành k phần (ví dụ: k=10), huấn luyện và đánh giá trên từng phần, cung cấp kết quả ổn định hơn.
* Leave One Out Cross Validation (LOOCV): Sử dụng LeaveOneOut để lấy từng mẫu làm tập kiểm tra, còn lại làm tập huấn luyện, phù hợp với dữ liệu nhỏ nhưng tốn tài nguyên.
* Repeated Random Test-Train Splits: Sử dụng ShuffleSplit để thực hiện nhiều lần chia ngẫu nhiên, kết hợp ưu điểm của train-test split và cross-validation.

**Cách thực hiện trong scikit-learn:**

* Sử dụng train\_test\_split để chia dữ liệu với test\_size=0.33 và random\_state=7 để tái lập kết quả.
* Sử dụng KFold(n\_splits=10) hoặc LeaveOneOut() để tạo các fold, sau đó dùng cross\_val\_score để đánh giá mô hình Logistic Regression.
* Sử dụng ShuffleSplit(n\_splits=10) để thực hiện chia ngẫu nhiên nhiều lần.
* Kết quả được báo cáo dưới dạng độ chính xác trung bình và độ lệch chuẩn (ví dụ: K-fold đạt 77.086% ± 5.091%).

**Lưu ý về đánh giá mô hình:**

* Độ lệch chuẩn cao (như trong LOOCV: 42.196%) cho thấy hiệu suất mô hình có thể không ổn định.
* Cần cân nhắc giữa độ chính xác và chi phí tính toán khi chọn phương pháp (LOOCV tốn kém hơn K-fold).
* Đảm bảo sử dụng cùng một random\_state để so sánh công bằng giữa các phương pháp.

**Chương 13: Compare Machine Learning Algorithms**

**Tầm quan trọng của việc so sánh thuật toán:**

* Hiểu rằng mỗi thuật toán học máy có ưu, nhược điểm riêng, và việc so sánh giúp chọn mô hình phù hợp nhất với bài toán cụ thể.
* So sánh nhất quán đảm bảo đánh giá công bằng, tránh thiên vị do cách chia dữ liệu hoặc tham số ngẫu nhiên.

**Các thuật toán được so sánh:**

* Logistic Regression (LR): Mô hình tuyến tính, đạt độ chính xác trung bình 77.087%.
* Linear Discriminant Analysis (LDA): Phân tích tuyến tính, đạt 76.697%.
* K-Nearest Neighbors (KNN): Dựa trên khoảng cách, đạt 71.099%, kém hơn các mô hình khác.
* Decision Tree Classifier (CART): Cây quyết định, đạt 70.455%, dễ bị quá khớp.
* Gaussian Naive Bayes (NB): Dựa trên xác suất, đạt 75.914%.
* Support Vector Machine (SVM): Phân loại phi tuyến, đạt 76.046%.

**Cách thực hiện trong scikit-learn và matplotlib:**

* Sử dụng KFold(n\_splits=10, random\_state=7) để đánh giá từng thuật toán bằng cross\_val\_score với tiêu chí scoring='accuracy'.
* Lưu kết quả độ chính xác của từng fold vào danh sách và tính trung bình, độ lệch chuẩn.
* Vẽ biểu đồ boxplot bằng matplotlib.pyplot.boxplot để trực quan hóa phân phối độ chính xác, giúp so sánh hiệu suất và độ ổn định của các thuật toán.

**Lưu ý về so sánh thuật toán:**

* Logistic Regression và LDA có hiệu suất tốt và ổn định nhất trên tập dữ liệu diabetes.csv.
* KNN và CART có độ chính xác thấp hơn, có thể do nhạy cảm với dữ liệu hoặc cần tinh chỉnh tham số.
* Biểu đồ boxplot giúp phát hiện thuật toán có độ biến động lớn (ví dụ: KNN với độ lệch chuẩn 0.051).
* Cần thử nghiệm thêm với các tham số hoặc tiền xử lý dữ liệu để cải thiện hiệu suất các thuật toán kém.