Machine Learning

1. Tìm ở các trang như Kaggle, Github, Google Colab, …. Tập dữ liệu thô về dự đoán (giá nhà, tuổi thọ, giá cổ phiếu, điểm,…).
2. Tiến hành phân tích tập dữ liệu
3. Tiến hành các bước tiền xử lý đã học

Nộp:

1. File dữ liệu thô .csv
2. File dữ liệu có thể dùng để huấn luyện .csv
3. File ipynb (2 code cell cuối để show 30 dòng của tập thô và 30 dòng của tập đã xử lý)

Google colab – Hướng dẫn cơ bản

1. Giới thiệu

- Google Colab (Colaboratory)
  - Môi trường lập trình Python trực tuyến của Google.
  - Viết, chạy, chia sẻ mã ngay trên trình duyệt, không cần cài đặt.
- Ưu điểm:
  - Miễn phí, dùng trên mọi hệ điều hành.
  - Hỗ trợ GPU/TPU tăng tốc học máy.
  - Tích hợp Google Drive, chia sẻ dễ như Google Docs.
- Định dạng: .ipynb – gồm Code Cell và Text Cell.

1. Thao tác: Truy cập

- 1. Truy cập trang: https://colab.research.google.com
- 2. Đăng nhập bằng tài khoản Google.
- 3. Chọn File → New Notebook để tạo một file mới.

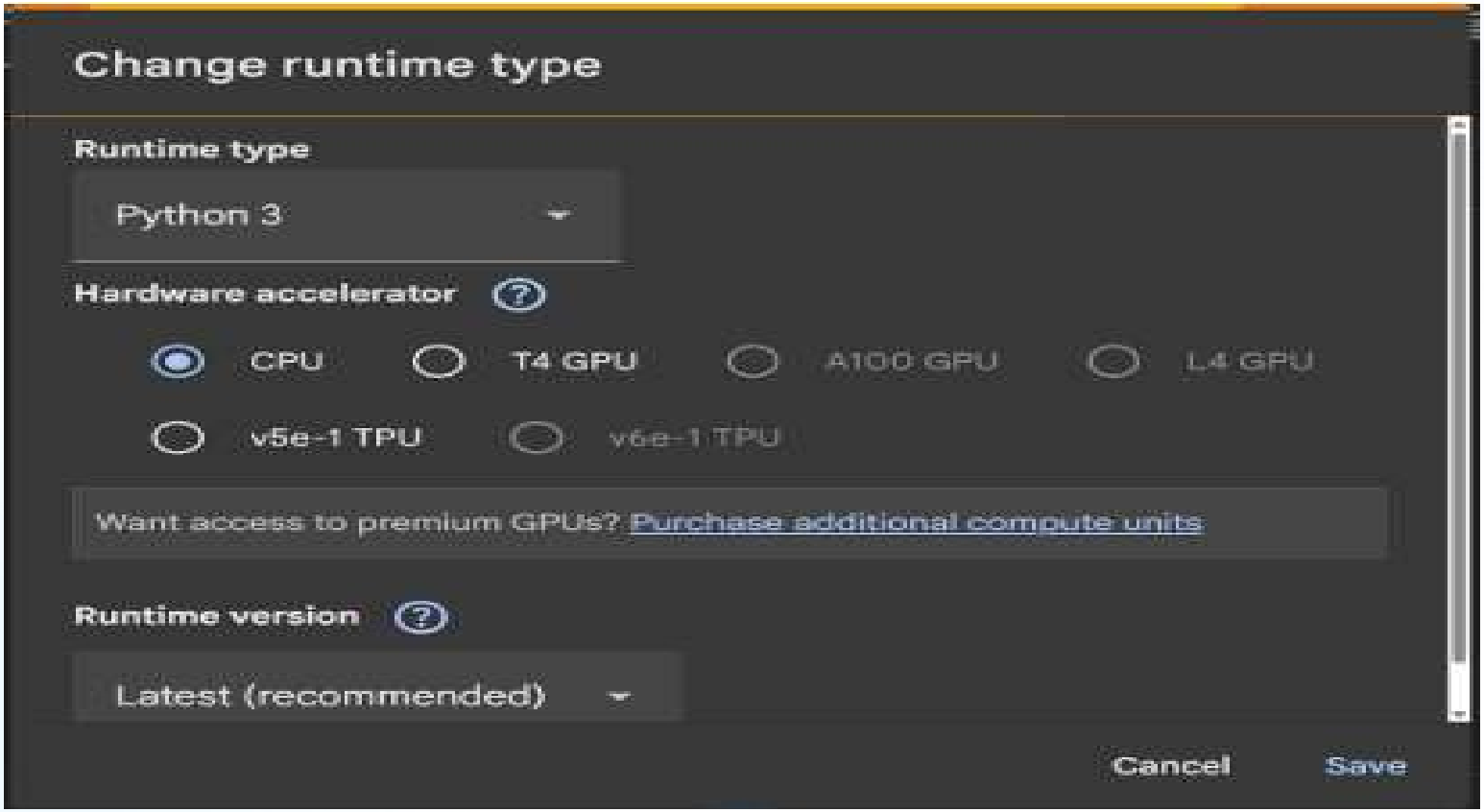
1. Thao tác: Đổi tên Notebook

- 1. Nhấn vào tiêu đề mặc định Untitled.ipynb ở góc trên cùng.
- 2. Gõ tên mới, ví dụ: Colab_Demo.ipynb.
- 3. Nhấn Enter để lưu tên mới.

1. Thao tác: Chọn loại Runtime

- 1. Chọn menu Runtime → Change runtime type.
- 2. Trong mục Hardware accelerator, chọn một trong hai:
    - None: chỉ sử dụng CPU.
    - GPU: dùng GPU (thường là NVIDIA Tesla T4).
- 3. Nhấn Save.

1. Thao tác: Chọn loại Runtime



1. Thao tác:

- Trong Google Colab, ta có hai cách phổ biến để truy cập dữ liệu hoặc mã nguồn:
- (1) tải trực tiếp file từ máy tính lên, hoặc
- (2) kết nối với Google Drive để sử dụng lâu dài.

1. **Thao tác:** Upload file từ máy (Local)

- Cách 1: Dùng giao diện — chọn Files → Upload ở thanh bên trái.
- Cách 2: Dùng code Python:

```
from google.colab import files
uploaded = files.upload()
```

1. **Thao tác:** Kết nối với Google Drive

- Dùng code Python:

```
from google.colab import drive
drive.mount('/content/drive')
```

1. **Thao tác:** Tải notebook hoặc file về máy / lưu vào Drive

- Tải notebook: File → Download → Download .ipynb hoặc .py.
- Tải file qua code: Có thể dùng trong trường hợp cần tải xuống các file sau khi huấn luyện (train).

```
from google.colab import files
files.download('filename.csv')
```

1. **Thao tác:** Chia sẻ & quyền truy cập

- Chọn Share ở góc phải → nhập email người nhận hoặc tạo link chia sẻ.
- Các mức quyền phổ biến:
  - Viewer: chỉ xem được notebook.
  - Commenter: có thể nhận xét.
  - Editor: có thể chỉnh sửa và chạy code.

QUY TRÌNH XÂY DỰNG MỘT DỰ ÁN HỌC MÁY HOÀN CHỈNH

Quy trình

1. Xác định Mục tiêu và Phạm vi Dự án (Project Definition)
2. Thu thập và Tích hợp Dữ liệu (Data Acquisition)
3. Tiền xử lý Dữ liệu (Data Preprocessing)
4. Phân tích Dữ liệu Khám phá (Exploratory Data Analysis - EDA)
5. Xây dựng và Huấn luyện Mô hình (Modeling and Training)
6. Đánh giá Mô hình (Model Evaluation)
7. Triển khai và Giám sát (Deployment and Monitoring)

1. Xác định Mục tiêu và Phạm vi Dự án (Project Definition)

Đây là bước nền tảng quan trọng nhất.

Xác định mục tiêu:

Vấn đề kinh doanh hoặc nghiên cứu cụ thể cần được giải quyết là gì?

Ví dụ: Giảm tỷ lệ khách hàng bỏ đi, Phân loại hình ảnh y tế, Dự đoán giá nhà.

1. Xác định Mục tiêu và Phạm vi Dự án (Project Definition)

Đây là bước nền tảng quan trọng nhất.

Xác định T:

Mô hình cần thực hiện nhiệm vụ gì?

Phân loại (Classification),

Hồi quy (Regression),

Phân cụm (Clustering),

….

1. Xác định Mục tiêu và Phạm vi Dự án (Project Definition)

Đây là bước nền tảng quan trọng nhất.

Xác định E: Mô hình cần kinh nghiệm gì?

Học có giám sát

Học không có giám sát

Học bán giám sát

hay Học tăng cường (Reinforcement Learning).

1. Xác định Mục tiêu và Phạm vi Dự án (Project Definition)

Đây là bước nền tảng quan trọng nhất.
Xác định độ đo P: Định lượng thành công bằng cách nào? Xác định các chỉ số đo lường chính (KPIs) như độ chính xác (Accuracy), F1-Score, RMSE, hoặc lợi nhuận kinh doanh mong muốn.

2. Thu thập và Tích hợp Dữ liệu (Data Acquisition)

Dữ liệu là "nguyên liệu" cho mô hình ML.
Nguồn Dữ liệu: Thu thập dữ liệu từ các nguồn nội bộ (database, log file), nguồn công khai (Kaggle, Data.gov) hoặc qua API.

Tích hợp: Kết hợp dữ liệu từ nhiều nguồn khác nhau vào một kho dữ liệu duy nhất.

Kiểm tra Chất lượng: Kiểm tra sơ bộ các vấn đề về thiếu sót, trùng lặp, hoặc không nhất quán trong dữ liệu.

3. Tiền xử lý Dữ liệu (Data Preprocessing)

Dữ liệu thô không thể đưa trực tiếp vào mô hình.

Xử lý Dữ liệu Khuyết thiếu:

Xử lý Dữ liệu Ngoại lai (Outliers):

Mã hóa Dữ liệu (Encoding):

Chuẩn hóa/Tỷ lệ hóa (Normalization/Scaling):

3. Tiền xử lý Dữ liệu (Data Preprocessing)

Xử lý Dữ liệu Khuyết thiếu: Điền vào (Imputation) bằng giá trị trung bình/trung vị/mode, hoặc loại bỏ các hàng/cột chứa quá nhiều giá trị khuyết.

PatientID	Age	BloodPressure	Gender	Diabetes
1	25	120	M	0
2	NaN	130	F	1
3	40	NaN	F	0
4	35	140	NaN	0

3. Tiền xử lý Dữ liệu (Data Preprocessing)

Xử lý Dữ liệu Ngoại lai (Outliers): Phát hiện và xử lý các giá trị bất thường (ví dụ: dùng Z-score hoặc IQR).

StudentID	Math	Physics	Chemistry
1	85	90	78
2	88	95	82
3	92	89	80
4	87	93	79
5	250	91	77
6	90	94	81

3. Tiền xử lý Dữ liệu (Data Preprocessing)

Mã hóa Dữ liệu (Encoding): Chuyển đổi dữ liệu phân loại (Categorical Data) thành định dạng số (ví dụ: One-Hot Encoding, Label Encoding).

StudentID	Gender	Major	Grade
1	M	Electrical	85
2	F	Mechanical	90
3	F	Electrical	78
4	M	Civil	88
5	M	Mechanical	92

StudentID	Grade	Gender_encoded	Major_encoded
1	85	1	1
2	90	0	2
3	78	0	1
4	88	1	0
5	92	1	2

3. Tiền xử lý Dữ liệu (Data Preprocessing)

Chuẩn hóa/Tỷ lệ hóa (Normalization/Scaling): Đưa các đặc trưng về cùng một phạm vi để mô hình học hiệu quả hơn (ví dụ: MinMaxScaler, StandardScaler).

HouseID	Area (m²)	Bedrooms	Age (years)	Price ($1000)
1	120	3	10	300
2	85	2	5	200
3	150	4	20	400
4	95	2	8	250
5	200	5	15	500

Area	Bedrooms	Age
-0.30	0.0	-0.20
-1.10	-1.0	-1.38
0.51	1.0	1.45
-0.88	-1.0	-0.71
1.77	1.0	0.83

4. Phân tích Dữ liệu Khám phá (Exploratory Data Analysis - EDA)

Mục đích là hiểu sâu hơn về dữ liệu và mối quan hệ giữa các biến.

Thống kê Mô tả:

Trực quan hóa Dữ liệu:

Phân tích Tương quan:

Lựa chọn Đặc trưng (Feature Selection):

Kỹ thuật Đặc trưng (Feature Engineering):

4. Phân tích Dữ liệu Khám phá (Exploratory Data Analysis - EDA)

Mục đích là hiểu sâu hơn về dữ liệu và mối quan hệ giữa các biến.
Thống kê Mô tả: Tính toán trung bình, độ lệch chuẩn, phân vị cho các đặc trưng.

4. Phân tích Dữ liệu Khám phá

Thống kê Mô tả:

StudentID	Hours_Studied	Attendance	Previous_Grade	Final_Grade
1	10	80	NaN	88
2	8	90	78	82
3	25	500	92	95
4	6	70	65	68
5	NaN	95	170	92

4. Phân tích Dữ liệu Khám phá

Thống kê Mô tả:

StudentID	Hours_Studied	Attendance (%)	Previous_Grade	Final_Grade
1	10	80	NaN	88
2	8	90	78	82
3	25	500	92	95
4	6	70	65	68
5	NaN	95	170	92

	Hours_Studied	Attendance	Previous_Grade	Final_Grade
count	5.0	5.0	5.0	5.0
mean	10.0	84.0	81.6	85.0
std	3.162	9.27	11.09	11.97
min	6.0	70.0	65.0	68.0
25%	8.0	80.0	78.0	82.0
50%	10.0	85.0	85.0	88.0
75%	12.0	90.0	88.0	92.0
max	14.0	95.0	92.0	95.0

4. Phân tích Dữ liệu Khám phá (Exploratory Data Analysis - EDA)

Mục đích là hiểu sâu hơn về dữ liệu và mối quan hệ giữa các biến.
Trực quan hóa Dữ liệu: Sử dụng biểu đồ histogram, scatter plot, box plot để xem xét phân phối và mối quan hệ giữa các biến.

4. Phân tích Dữ liệu Khám phá

Trực quan hóa Dữ liệu:

- Biểu đồ Histogram
- Biểu đồ Scatter
- Boxplot

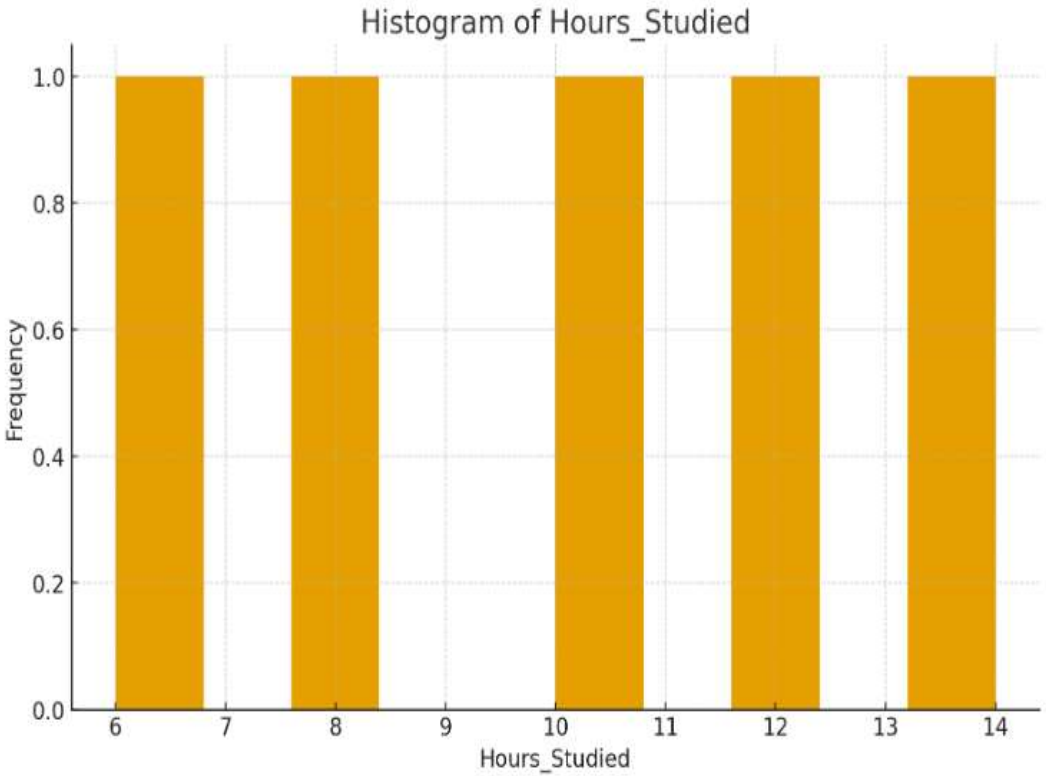
4. Phân tích Dữ liệu Khám phá

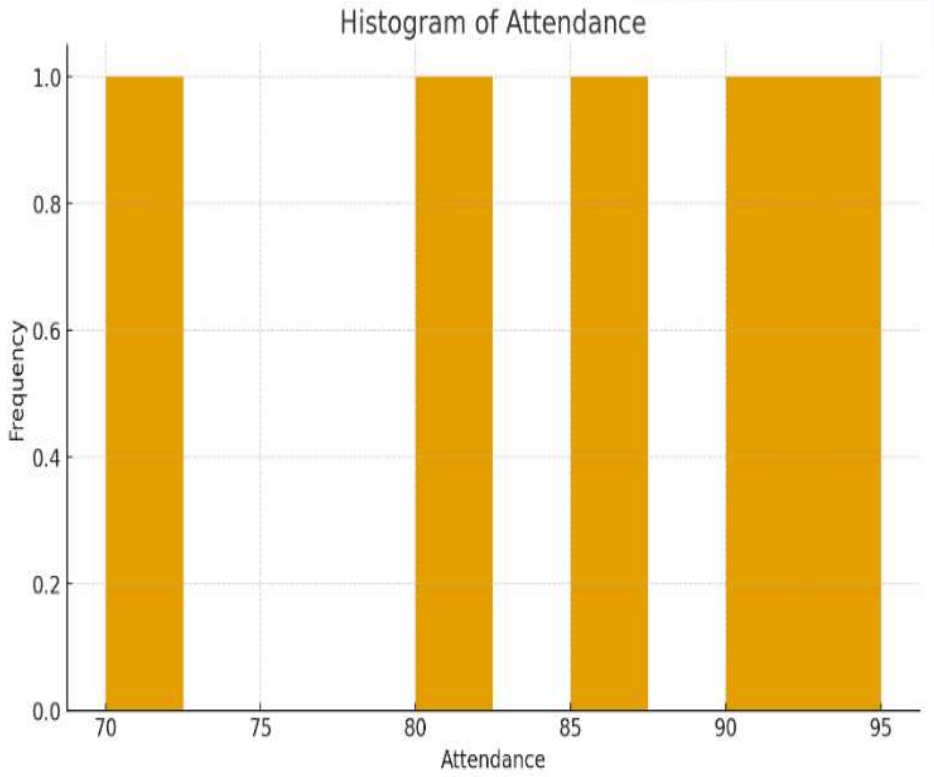
Trực quan hóa Dữ liệu:

- Biểu đồ Histogram
  cho thấy **phân bố giá trị** của từng biến
- Biểu đồ Scatter
  mối quan hệ với target (label y)
- Boxplot
  phát hiện giá trị bất thường

4. Phân tích Dữ liệu Khám phá

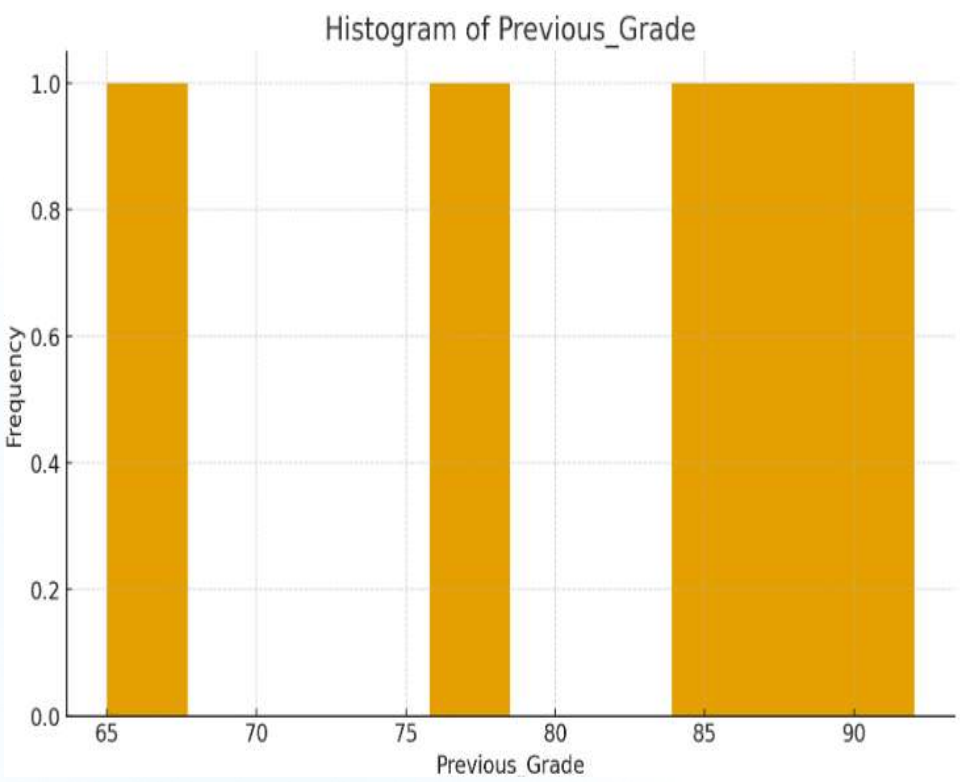
Biểu đồ Histogram:

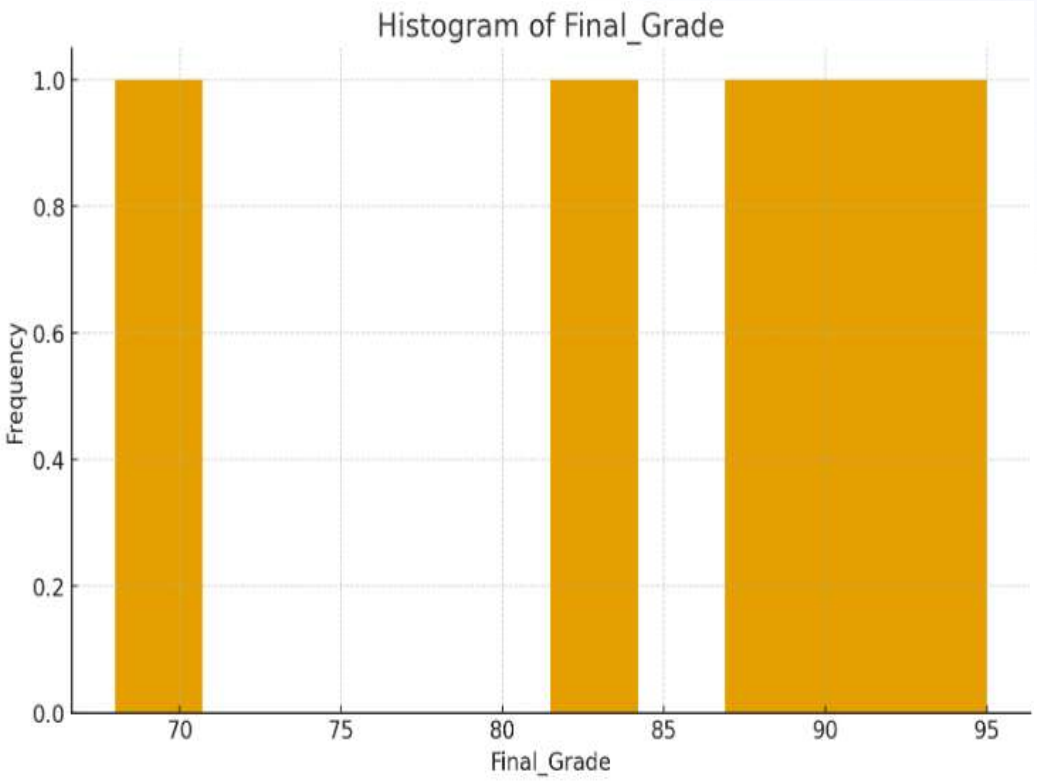




4. Phân tích Dữ liệu Khám phá

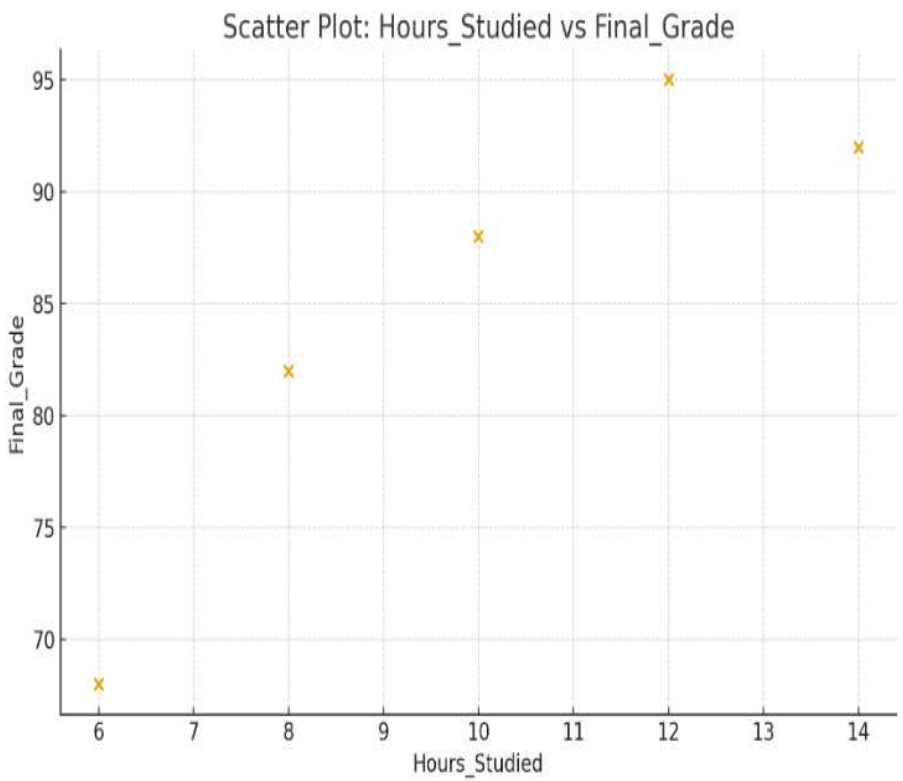
Biểu đồ Histogram:

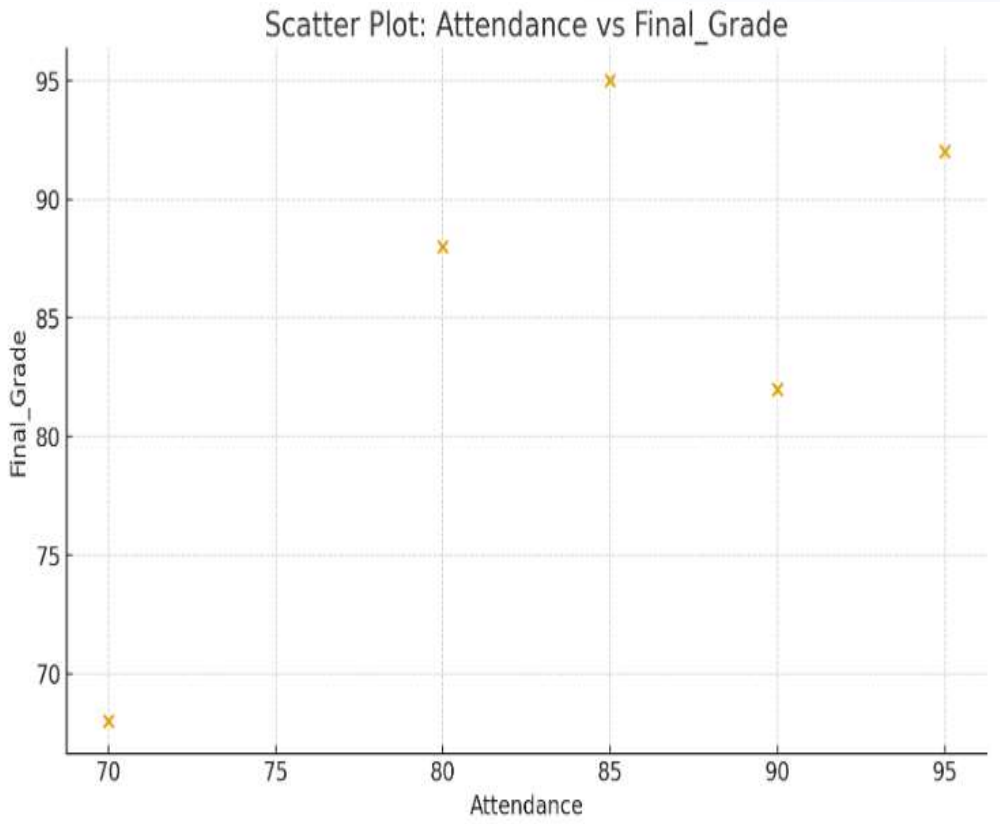




4. Phân tích Dữ liệu Khám phá

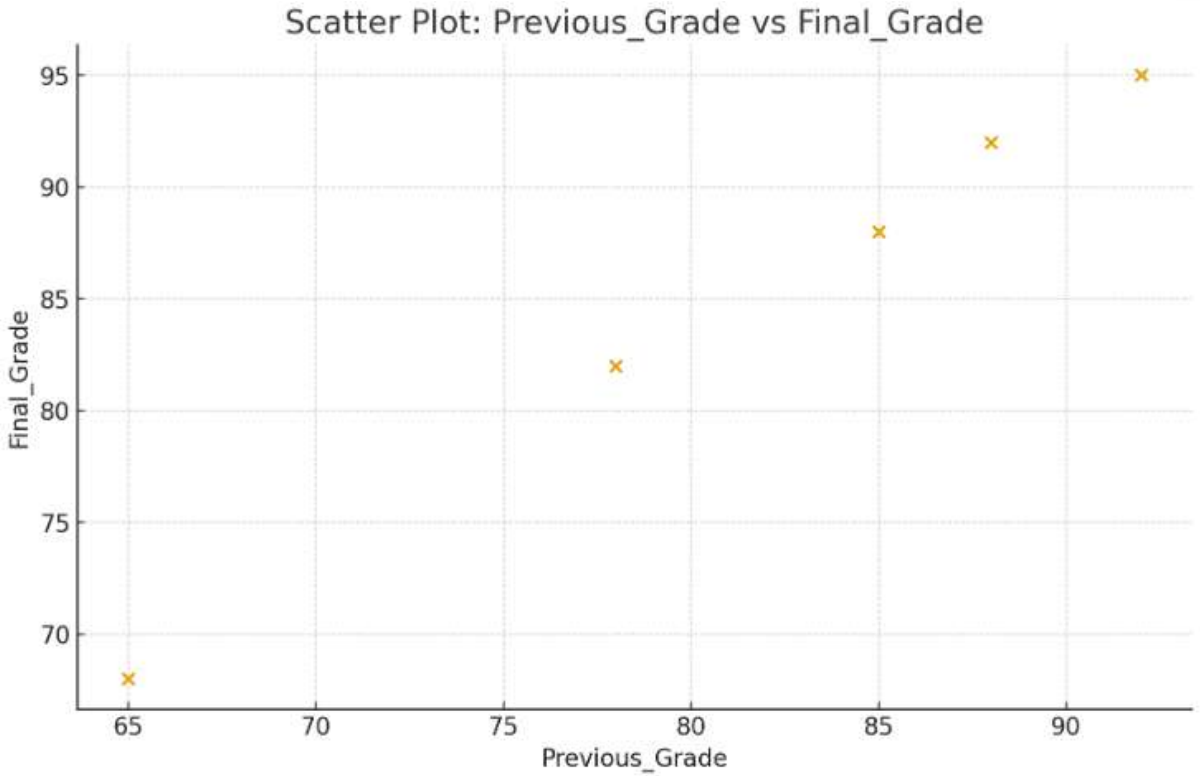
Biểu đồ Scatter





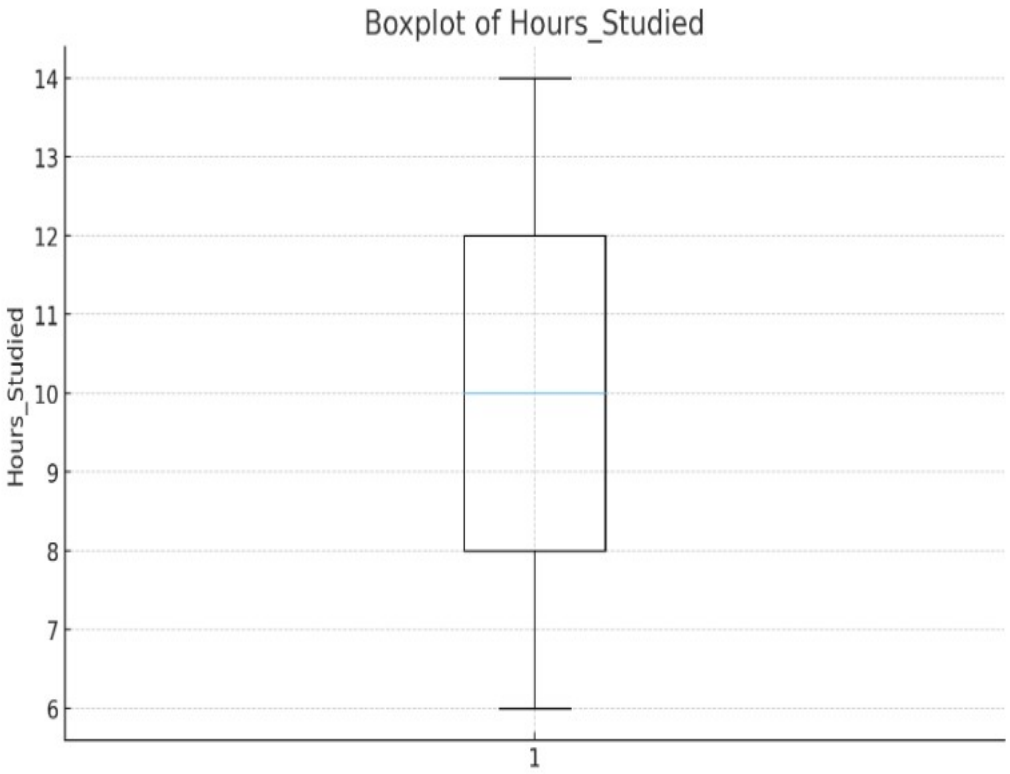
4. Phân tích Dữ liệu Khám phá

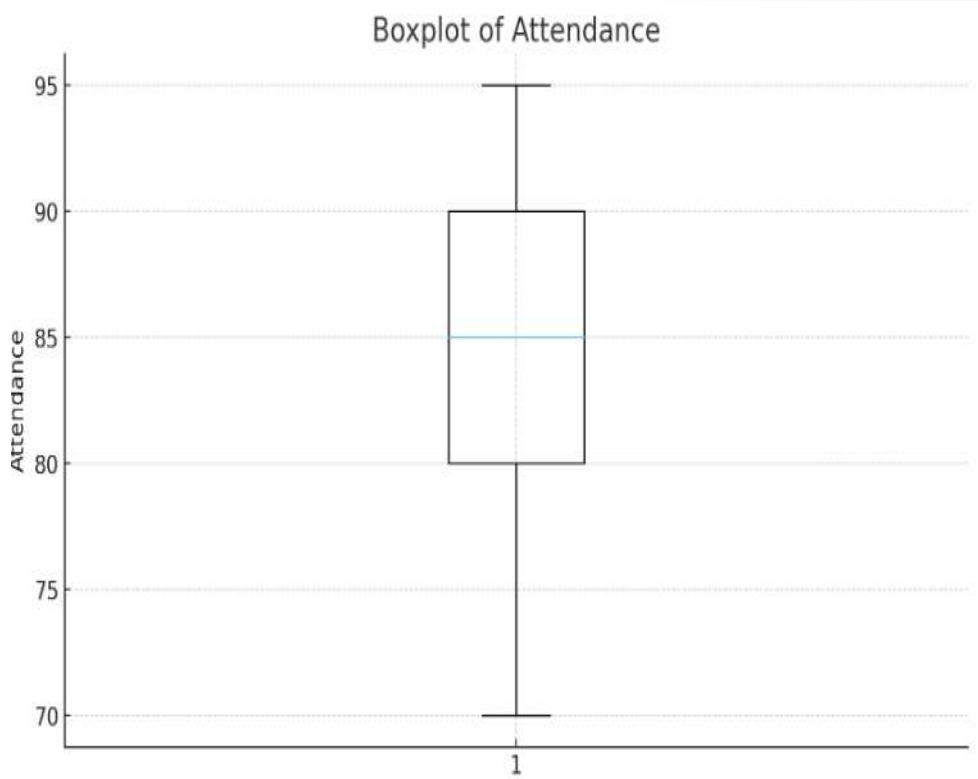
Biểu đồ Scatter



4. Phân tích Dữ liệu Khám phá

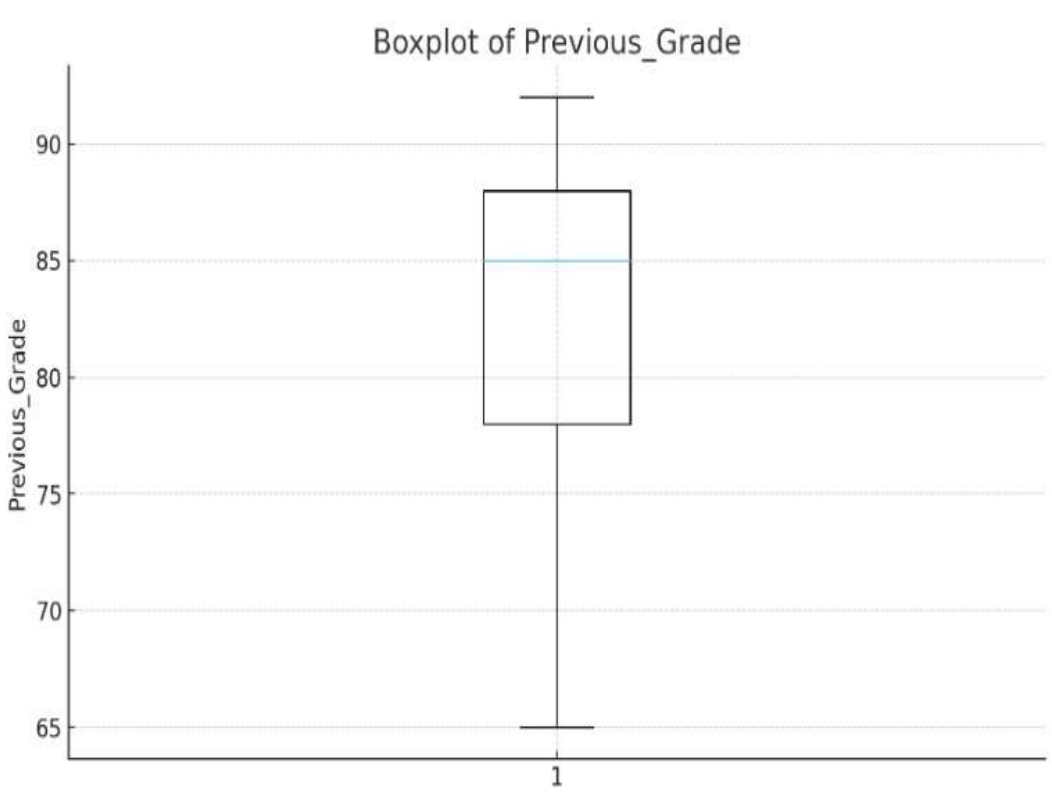
Boxplot

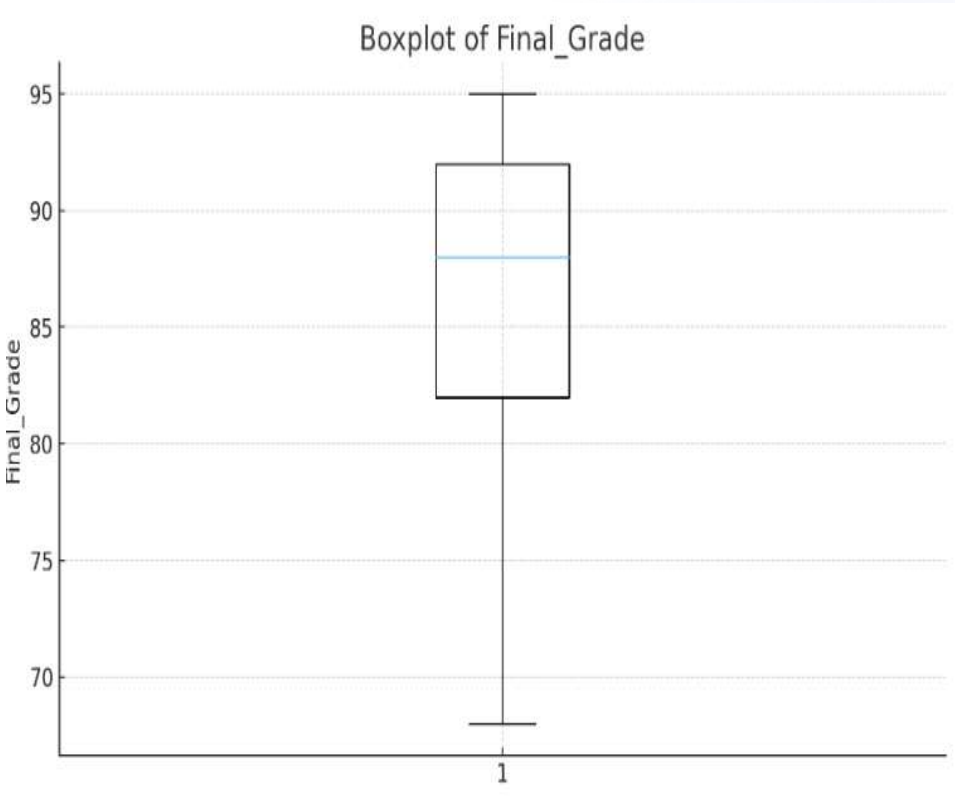




4. Phân tích Dữ liệu Khám phá

Boxplot



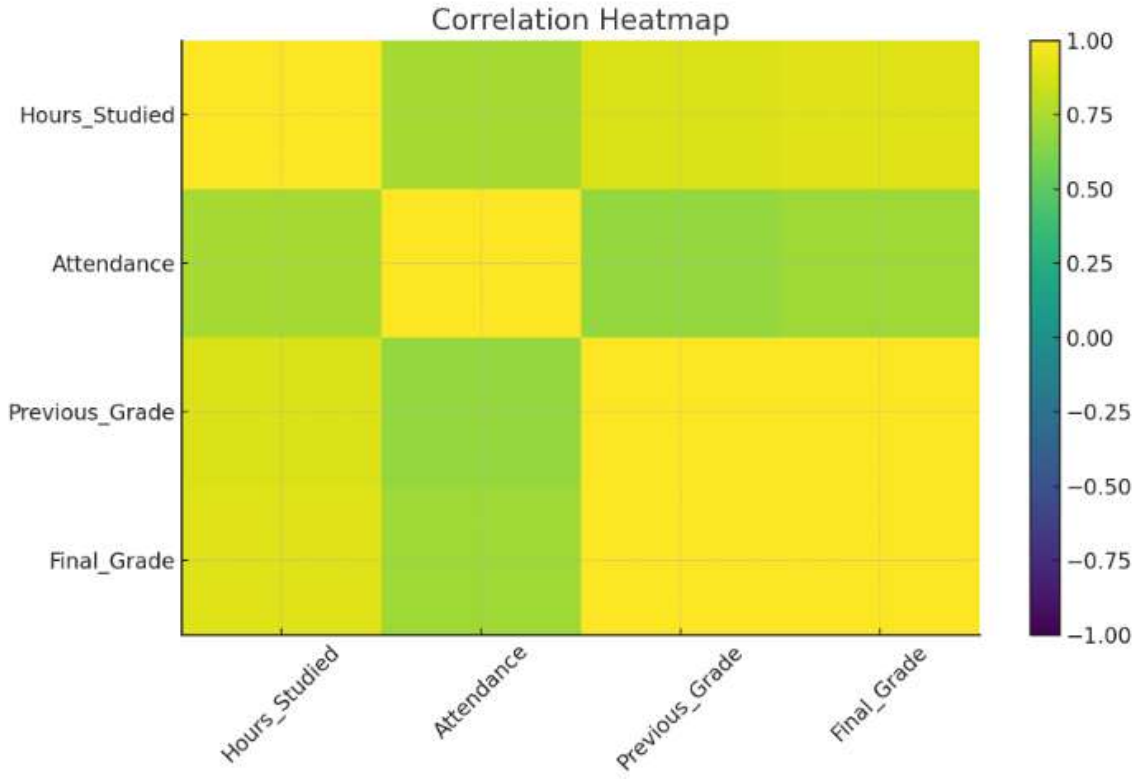


4. Phân tích Dữ liệu Khám phá (Exploratory Data Analysis - EDA)

Phân tích Tương quan: Tìm kiếm mối quan hệ tuyến tính giữa các đặc trưng và giữa đặc trưng với biến mục tiêu.

4. Phân tích Dữ liệu Khám phá

Phân tích Tương quan (Heatmap Correlation)**:**



Yếu tố	Mức ảnh hưởng đến điểm cuối kỳ
Số giờ học	**Mạnh nhất**
Điểm nền học kỳ trước	**Mạnh**
Chuyên cần	**Vừa**

4. Phân tích Dữ liệu Khám phá (Exploratory Data Analysis - EDA)

Lựa chọn Đặc trưng (Feature Selection): Quyết định những đặc trưng nào có ý nghĩa nhất đối với mô hình.
Kỹ thuật Đặc trưng (Feature Engineering): Tạo ra các đặc trưng mới từ đặc trưng hiện có để tăng cường khả năng dự đoán của mô hình.

5. Xây dựng và Huấn luyện Mô hình (Modeling and Training)

Đây là giai đoạn tạo ra mô hình dự đoán.

Chia Dữ liệu: Chia tập dữ liệu thành ba phần: Tập Huấn luyện (Training), Tập Thẩm định (Validation) và Tập Kiểm tra (Test).

5. Xây dựng và Huấn luyện Mô hình (Modeling and Training)

Đây là giai đoạn tạo ra mô hình dự đoán.

Chia Dữ liệu: Chia tập dữ liệu thành ba phần: Tập Huấn luyện (Training), Tập Thẩm định (Validation) và Tập Kiểm tra (Test).

Lựa chọn Thuật toán: Chọn thuật toán phù hợp với mục tiêu (ví dụ: Linear Regression cho hồi quy, Random Forest hoặc Neural Networks cho phân loại).

5. Xây dựng và Huấn luyện Mô hình (Modeling and Training)

Đây là giai đoạn tạo ra mô hình dự đoán.

Chia Dữ liệu: Chia tập dữ liệu thành ba phần: Tập Huấn luyện (Training), Tập Thẩm định (Validation) và Tập Kiểm tra (Test).

Lựa chọn Thuật toán: Chọn thuật toán phù hợp với mục tiêu (ví dụ: Linear Regression cho hồi quy, Random Forest hoặc Neural Networks cho phân loại).

Huấn luyện: Đưa tập huấn luyện vào mô hình và điều chỉnh trọng số (weights).

5. Xây dựng và Huấn luyện Mô hình (Modeling and Training)

Đây là giai đoạn tạo ra mô hình dự đoán.

Chia Dữ liệu: Chia tập dữ liệu thành ba phần: Tập Huấn luyện (Training), Tập Thẩm định (Validation) và Tập Kiểm tra (Test).

Lựa chọn Thuật toán: Chọn thuật toán phù hợp với mục tiêu (ví dụ: Linear Regression cho hồi quy, Random Forest hoặc Neural Networks cho phân loại).

Huấn luyện: Đưa tập huấn luyện vào mô hình và điều chỉnh trọng số (weights).

Tinh chỉnh Siêu tham số (Hyperparameter Tuning): Tối ưu hóa các tham số bên ngoài mô hình (ví dụ: tốc độ học, số lượng cây trong rừng) bằng các kỹ thuật như Grid Search hoặc Randomized Search.

6. Đánh giá Mô hình (Model Evaluation)

Đánh giá hiệu suất của mô hình trên dữ liệu chưa từng thấy.
Đánh giá trên Tập Thẩm định: Sử dụng tập thẩm định để điều chỉnh siêu tham số và tránh hiện tượng quá khớp (Overfitting).
Đánh giá trên Tập Kiểm tra: Dùng tập kiểm tra (chỉ dùng một lần) để xác định hiệu suất cuối cùng và khách quan của mô hình.
Phân tích Kết quả: Kiểm tra các chỉ số KPIs đã xác định ở Bước 1. (Ví dụ: Ma trận nhầm lẫn (Confusion Matrix), ROC curve, Precision, Recall).
So sánh Mô hình: So sánh hiệu suất của mô hình đã chọn với các mô hình thay thế hoặc mô hình cơ sở (Baseline Model).

7. Triển khai và Giám sát (Deployment and Monitoring)

Đưa mô hình vào môi trường thực tế để tạo ra giá trị.
Đóng gói Mô hình (Packaging): Lưu mô hình đã huấn luyện (ví dụ: dùng thư viện pickle hoặc joblib).
Triển khai (Deployment): Tích hợp mô hình vào ứng dụng web, dịch vụ đám mây (AWS, GCP, Azure), hoặc thiết bị biên.
Giám sát (Monitoring): Theo dõi hiệu suất của mô hình trong thời gian thực.
Drift (Trôi dạt dữ liệu): Theo dõi sự thay đổi trong phân phối dữ liệu đầu vào.
Hiệu suất: Đảm bảo độ chính xác (hoặc các KPIs khác) không bị suy giảm theo thời gian.
Tái Huấn luyện (Retraining): Lập kế hoạch và quy trình để tái huấn luyện mô hình khi hiệu suất giảm sút do dữ liệu bị trôi dạt.

Tải Dữ Liệu

Nạp dữ liệu thô vào hệ thống từ nhiều nguồn khác nhau.

2.2 Tải tập dữ liệu

Vấn đề: Bạn muốn load một bộ dữ liệu mô phỏng với các thuộc tính cụ thể.

Giải pháp: scikit-learn cung cấp 03 dữ liệu.

- load_boston: 503 quan sát về giá nhà ở Boston.
- load_iris: 150 quan sát về các phép đo hoa Iris.
- load_digits:1.797 quan sát từ hình ảnh chữ số viết tay.

2.2 Tải tập dữ liệu

```
# Load scikit-learn's datasets
from sklearn import datasets

# Load digits dataset
digits = datasets.load_digits()

# Create features matrix
features = digits.data
```

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13.,
       15., 10., 15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,
        8.,  0.,  0.,  4., 12.,  0.,  0.,  8.,  8.,  0.,  0.,
        5.,  8.,  0.,  0.,  9.,  8.,  0.,  0.,  4., 11.,  0.,
        1., 12.,  7.,  0.,  0.,  2., 14.,  5., 10., 12.,  0.,
        0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

```
# Create target vector
target = digits.target

# View first observation
features[0]
```

2.2 Tạo tập dữ liệu mô phỏng

Vấn đề: Bạn cần tạo ra một bộ dữ liệu mô phỏng với các thuộc tính cụ thể.

Giải pháp: scikit-learn cung cấp nhiều phương thức để tạo dữ liệu.

Các hàm chính:

make_regression: Tạo dữ liệu cho các bài toán hồi quy tuyến tính.

make_classification: Tạo dữ liệu cho các bài toán phân loại.

make_blobs: Tạo dữ liệu với các cụm (blobs), hữu ích cho việc thử nghiệm phân cụm.

2.2 Tạo tập dữ liệu mô phỏng

```
# Load library
from sklearn.datasets import make_regression

# Generate features matrix, target vector, and the true coefficients
features, target, coefficients = make_regression(n_samples = 100,
                                                 n_features = 3,
                                                 n_informative = 3,
                                                 n_targets = 1,
                                                 noise = 0.0,
                                                 coef = True,
                                                 random_state = 1)

# View feature matrix and target vector
print('Feature Matrix\n', features[:3])
print('Target Vector\n', target[:3])
```

```
Feature Matrix
 [[ 1.29322588 -0.61736206 -0.11044703]
 [-2.793085    0.36633201  1.93752881]
 [ 0.80186103 -0.18656977  0.0465673 ]]
Target Vector
 [-10.37865986  25.5124503   19.67705609]
```

2.2 Tạo tập dữ liệu mô phỏng

```
# Load library
from sklearn.datasets import make_classification

# Generate features matrix and target vector
features, target = make_classification(n_samples = 100,
                                       n_features = 3,
                                       n_informative = 3,
                                       n_redundant = 0,
                                       n_classes = 2,
                                       weights = [.25, .75],
                                       random_state = 1)

# View feature matrix and target vector
print('Feature Matrix\n', features[:3])
print('Target Vector\n', target[:3])
```

```
Feature Matrix
 [[ 1.06354768 -1.42632219  1.02163151]
 [ 0.23156977  1.49535261  0.33251578]
```

2.2 Tạo tập dữ liệu mô phỏng

```
# Load library
from sklearn.datasets import make_blobs

# Generate feature matrix and target vector
features, target = make_blobs(n_samples = 100,
                              n_features = 2,
                              centers = 3,
                              cluster_std = 0.5,
                              shuffle = True,
                              random_state = 1)

# View feature matrix and target vector
print('Feature Matrix\n', features[:3])
print('Target Vector\n', target[:3])
```

```
Feature Matrix
 [[ -1.22685609   3.25572052]
 [ -9.57463218  -4.38310652]
 [-10.71976941  -4.20558148]]
Target Vector
 [0 1 1]
```

2.3 Tải tệp CSV

- **Vấn đề:** Bạn cần nhập tệp dữ liệu dạng CSV.
- **Giải pháp:** Sử dụng hàm read_csv của thư viện pandas.
- **Ví dụ:**
  - import pandas as pd
  - dataframe = pd.read_csv('du_lieu.csv')
- **Tham số hữu ích:**
  - sep: Chỉ định ký tự phân tách (ví dụ: sep=',' hoặc sep='\t').
  - header: Chỉ định hàng nào là tiêu đề (ví dụ: header=0 hoặc header=None).

2.4 Tải tệp Excel

- **Vấn đề:** Bạn cần nhập dữ liệu từ một tệp Excel.
- **Giải pháp:** Sử dụng hàm read_excel của pandas.
- **Ví dụ:**
  - import pandas as pd
  - dataframe = pd.read_excel('du_lieu.xlsx')
- **Tham số hữu ích:**
  - sheet_name: Chỉ định sheet cần tải (theo tên hoặc chỉ số, ví dụ: sheet_name=0 hoặc sheet_name='Doanh_thu').

2.5 Tải tệp JSON

- **Vấn đề:** Bạn cần tải tệp JSON để tiền xử lý.
- **Giải pháp:** Sử dụng hàm read_json của pandas.
- **Ví dụ:**
  - import pandas as pd
  - dataframe = pd.read_json('du_lieu.json')
- **Tham số hữu ích:**
  - orient: Chỉ định cấu trúc của tệp JSON (ví dụ: 'split', 'records', 'index', 'columns').
- Hàm json_normalize cũng rất hữu ích để làm phẳng dữ liệu JSON bán cấu trúc.

2.6 Truy vấn Cơ sở dữ liệu SQL

- **Vấn đề:** Bạn cần tải dữ liệu từ cơ sở dữ liệu (CSDL) bằng SQL.

- **Giải pháp:** Sử dụng read_sql_query của pandas và một engine kết nối (ví dụ: SQLAlchemy).
- **Ví dụ:**
  - from sqlalchemy import create_engine
  - database_connection = create_engine('sqlite:///sample.db')
  - dataframe = pd.read_sql_query('SELECT * FROM data', database_connection)

Thao tác dữ liệu

Biến đổi dữ liệu thô sang định dạng sạch sẽ, có tổ chức (Data Wrangling).

3.1 & 3.2: Tạo & Mô tả DataFrame

- **3.1 Tạo DataFrame:**
  - ▶Vấn đề: Tạo một DataFrame mới.
  - ▶Giải pháp: Sử dụng pd.DataFrame() và thêm cột hoặc hàng.
  - ▶Ví dụ: df = pd.DataFrame(), df['Name'] = ['Jack', 'Steve']
- **3.2 Mô tả Dữ liệu:**
  - ▶Vấn đề: Xem các đặc điểm của DataFrame.
  - ▶Giải pháp: Sử dụng df.head(2) (xem 2 hàng đầu),
    df.shape (xem kích thước),
    df.describe() (xem thống kê mô tả).

3.3 & 3.4: Điều hướng & Lọc

- **3.3 Điều hướng DataFrames:**
  - ▶Vấn đề: Chọn các hàng hoặc cột cụ thể.
  - ▶iloc (Vị trí): df.iloc[0] (hàng đầu tiên), df.iloc[1:4] (hàng 1, 2, 3).
  - ▶loc (Nhãn): df.loc['Allen, Miss Elisabeth'] (sau khi set_index).
- **3.4 Lựa chọn theo điều kiện:**
  - ▶Vấn đề: Chọn các hàng dựa trên một điều kiện.
  - ▶Một điều kiện: df[df['Sex'] == 'female']
  - ▶Nhiều điều kiện: df[(df['Sex'] == 'female') & (df['Age'] >= 65)]

3.5 & 3.6: Sửa đổi DataFrame

- **3.5 Thay thế giá trị:**
  - ▶Vấn đề: Thay thế các giá trị trong DataFrame.
  - ▶Giải pháp: df['Sex'].replace("female", "Woman").
  - ▶Cũng hỗ trợ thay thế nhiều giá trị và regex: df.replace(r"1st", "First", regex=True).
- **3.6 Đổi tên cột:**
  - ▶Vấn đề: Đổi tên một hoặc nhiều cột.
  - ▶Giải pháp: df.rename(columns={'PClass': 'Passenger Class'}).

3.7 & 3.8: Thống kê & Giá trị duy nhất

- **3.7 Thống kê cơ bản:**
  - ▶Vấn đề: Tìm min, max, sum, mean, count của một cột số.
  - ▶Giải pháp: df['Age'].max(), df['Age'].mean(), df['Age'].count(), v.v.
- **3.8 Giá trị duy nhất:**
  - ▶Vấn đề: Tìm các giá trị duy nhất trong một cột.
  - ▶df['Sex'].unique(): Trả về một mảng các giá trị duy nhất.
  - ▶df['Sex'].value_counts(): Trả về số lần xuất hiện của mỗi giá trị.
  - ▶df['PClass'].nunique(): Trả về số lượng giá trị duy nhất.

3.9 – 3.12: Xử lý dữ liệu thiếu & thừa

- **3.9 Xử lý giá trị thiếu:**
  - ▶Sử dụng df['Age'].isnull() để tìm các giá trị NaN.
- **3.10 Xóa cột:** Sử dụng df.drop('Age', axis=1).
- **3.11 Xóa hàng:** Sử dụng điều kiện boolean: df[df['Sex'] != 'male'].
- **3.12 Xóa hàng trùng lặp:**
  - ▶df.drop_duplicates(): Xóa các hàng trùng lặp hoàn toàn.
  - ▶df.drop_duplicates(subset=['Sex']): Chỉ giữ lại hàng đầu tiên cho mỗi giá trị 'Sex' duy nhất.

3.13 & 3.14: Nhóm dữ liệu

- **3.13 Nhóm theo giá trị:**
  - ▶Vấn đề: Nhóm các hàng theo một giá trị chung.
  - ▶Giải pháp: groupby() kết hợp với một hàm tổng hợp.
  - ▶Ví dụ: df.groupby('Sex').mean()
- **3.14 Nhóm theo thời gian:**
  - ▶Vấn đề: Nhóm các hàng theo các khoảng thời gian.
  - ▶Giải pháp: resample() trên cột index dạng datetime.
  - ▶Ví dụ: df.resample('W').sum() (Tổng theo tuần - Week).

3.15 – 3.17: Áp dụng các hàm

- **3.15 Lặp qua một cột:**
  - ▶Sử dụng for name in df['Name']: print(name). (Không hiệu quả)
- **3.16 Áp dụng hàm lên cột:**
  - ▶Cách tốt hơn là dùng apply().
  - ▶Ví dụ: df['Name'].apply(lambda x: x.upper())
- **3.17 Áp dụng hàm lên nhóm:**
  - ▶Kết hợp groupby() và apply().
  - ▶Ví dụ: df.groupby('Sex').apply(lambda x: x.count())

3.18 & 3.19: Kết hợp DataFrames

- **3.18 Nối (Concatenating):**
  - ▶Vấn đề: "Xếp chồng" các DataFrame.
  - ▶pd.concat([df_a, df_b], axis=0) (Nối theo hàng).
  - ▶pd.concat([df_a, df_b], axis=1) (Nối theo cột).
- **3.19 Trộn (Merging):**
  - ▶Vấn đề: "Join" các DataFrame giống như trong SQL.
  - ▶pd.merge(df_left, df_right, on='employee_id', how='outer')
  - ▶Tham số how: 'inner' (mặc định), 'outer', 'left', 'right'.

3.18 & 3.19: Kết hợp DataFrames

Tham số `how`	Ý nghĩa	Giữ khóa từ bảng trái	Giữ khóa từ bảng phải	Giá trị không khớp
`inner`	**Inner join** – chỉ giữ phần giao nhau	❌ Chỉ giữ nếu khớp	❌ Chỉ giữ nếu khớp	Bỏ hết
`left`	**Left join** – giữ toàn bộ khóa từ bảng trái	✅ Giữ toàn bộ	❌ Chỉ giữ nếu khớp	Điền NaN cho phần thiếu từ bảng phải
`right`	**Right join** – giữ toàn bộ khóa từ bảng phải	❌ Chỉ giữ nếu khớp	✅ Giữ toàn bộ	Điền NaN cho phần thiếu từ bảng trái
`outer`	**Full outer join** – giữ tất cả khóa từ cả hai bảng	✅ Giữ toàn bộ	✅ Giữ toàn bộ	Điền NaN cho phần thiếu từ bảng còn lại

Xử lý dữ liệu số

Biến đổi dữ liệu số thô thành các đặc trưng cho Machine Learning.

Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning, Chris Albon

4.1, 4.2 & 4.3: Các kỹ thuật co giãn

- **4.1 Co giãn (Rescaling):**
  - ▶Đưa các giá trị về một phạm vi, thường là [0, 1].
  - ▶Giải pháp: MinMaxScaler().
- **4.2 Chuẩn hóa (Standardizing):**
  - ▶Biến đổi đặc trưng để có trung bình =0 và độ lệch chuẩn =1.
  - ▶Giải pháp: StandardScaler().
- **4.3 Chuẩn hóa quan sát (Normalizing):**
  - ▶Co giãn các giá trị của từng *quan sát* (hàng) để có norm đơn vị (độ dài là 1)
  - ▶Giải pháp: Normalizer(norm='l2').

4.1, 4.2 & 4.3: Các kỹ thuật co giãn

```
# Load libraries
import numpy as np
from sklearn import preprocessing

# Create feature
feature = np.array([[-500.5],
                    [-100.1],
                    [0],
                    [100.1],
                    [900.9]])

# Create scaler
minmax_scale = preprocessing.MinMaxScaler(feature_range=(0, 1))

   # Scale feature
   scaled_feature = minmax_scale.fit_transform(feature)

   # Show feature
   scaled_feature

   array([[ 0.        ],
          [ 0.28571429],
          [ 0.35714286],
          [ 0.42857143],
          [ 1.        ]])
```

4.1, 4.2 & 4.3: Các kỹ thuật co giãn

```
# Load libraries
import numpy as np
from sklearn import preprocessing

# Create feature
x = np.array([[-1000.1],
              [-200.2],
              [500.5],
              [600.6],
              [9000.9]])

# Create scaler
scaler = preprocessing.StandardScaler()

# Transform the feature
standardized = scaler.fit_transform(x)

# Show feature
standardized
```

```
array([[-0.76058269],
       [-0.54177196],
       [-0.35009716],
       [-0.32271504],
       [ 1.97516685]])
```

4.1, 4.2 & 4.3: Các kỹ thuật co giãn

```
# Load libraries
import numpy as np
from sklearn.preprocessing import Normalizer

# Create feature matrix
features = np.array([[0.5, 0.5],
                     [1.1, 3.4],
                     [1.5, 20.2],
                     [1.63, 34.4],
                     [10.9, 3.3]])

# Create normalizer
normalizer = Normalizer(norm="l2")

# Transform feature matrix
normalizer.transform(features)
```

```
array([[ 0.70710678,  0.70710678],
       [ 0.30782029,  0.95144452],
       [ 0.07405353,  0.99725427],
       [ 0.04733062,  0.99887928],
       [ 0.95709822,  0.28976368]])
```

4.4 & 4.5: Biến đổi đặc trưng

- **4.4 Đặc trưng đa thức & tương tác:**
  - Vấn đề: Mô hình hóa các mối quan hệ phi tuyến tính và tương tác.
  - Giải pháp: PolynomialFeatures.
  - degree=2: Tạo ra $x_1, x_2, x_1^2, x_1x_2, x_2^2$.
  - interaction_only=True: Chỉ tạo x_1, x_2, x_1x_2.
- **4.5 Biến đổi đặc trưng tùy chỉnh:**
  - Vấn đề: Áp dụng một hàm tùy chỉnh (ví dụ: log, sin) cho các đặc trưng.
  - Giải pháp (scikit-learn): FunctionTransformer.
  - Giải pháp (pandas): df.apply(my_function).

4.6 & 4.7: Xử lý Outliers

- **4.6 Phát hiện Outliers:**

```
# Create one feature
feature = features[:,0]

# Create a function to return index of outliers
def indicies_of_outliers(x):
    q1, q3 = np.percentile(x, [25, 75])
    iqr = q3 - q1
    lower_bound = q1 - (iqr * 1.5)
    upper_bound = q3 + (iqr * 1.5)
    return np.where((x > upper_bound) | (x < lower_bound))

# Run function
indicies_of_outliers(feature)
```

```
(array([0]),)
```

4.6 & 4.7: Xử lý Outliers

- **4.7 Xử lý Outliers:**
  - **Chiến lược 1 (Xóa):** df[df['Bathrooms'] < 20].

```
# Load library
import pandas as pd

# Create DataFrame
houses = pd.DataFrame()
houses['Price'] = [534433, 392333, 293222, 4322032]
houses['Bathrcoms'] = [2, 3.5, 2, 116]
houses['Square_Feet'] = [1500, 2500, 1500, 48000]

# Filter observations
houses[houses['Bathrooms'] < 20]
```

	Price	Bathrooms	Square_Feet
0	534433	2.0	1500
1	392333	3.5	2500
2	293222	2.0	1500

4.6 & 4.7: Xử lý Outliers

- **4.7 Xử lý Outliers:**
  - **Chiến lược 2 (Đánh dấu):** Tạo đặc trưng nhị phân Is_Outlier.

```
# Load library
import numpy as np

# Create feature based on boolean condition
houses["Outlier"] = np.where(houses["Bathrooms"] < 20, 0, 1)

# Show data
houses
```

	Price	Bathrooms	Square_Feet	Outlier
0	534433	2.0	1500	0
1	392333	3.5	2500	0
2	293222	2.0	1500	0
3	4322032	116.0	48000	1

4.6 & 4.7: Xử lý Outliers

- **4.7 Xử lý Outliers:**
  - **Chiến lược 3 (Biến đổi):** Dùng np.log() để giảm ảnh hưởng.

```
# Log feature
houses["Log_Of_Square_Feet"] = [np.log(x) for x in houses["Square_Feet"]]

# Show data
houses
```

	Price	Bathrooms	Square_Feet	Outlier	Log_Of_Square_Feet
0	534433	2.0	1500	0	7.313220
1	392333	3.5	2500	0	7.824046
2	293222	2.0	1500	0	7.313220
3	4322032	116.0	48000	1	10.778956

4.8 & 4.9: Rời rạc hóa & Phân cụm

- **4.8 Rời rạc hóa (Discretization):**
  - Vấn đề: Chia đặc trưng số thành các thùng (bins) rời rạc.
  - Giải pháp 1 (Hai thùng): Binarizer(threshold=18).
  - Giải pháp 2 (Nhiều thùng): np.digitize(age, bins=[20, 30, 64]).
- **4.9 Nhóm quan sát bằng Phân cụm:**
  - Vấn đề: Nhóm các quan sát tương tự nhau.
  - Giải pháp: Sử dụng KMeans như một bước tiền xử lý.
  - Ví dụ: clusterer = KMeans(n_clusters=3), df['group'] = clusterer.fit_predict(features)

4.10 & 4.11: Xử lý giá trị số bị thiếu

- **4.10 Xóa quan sát:**
  - ▶Vấn đề: Cần xóa các hàng có giá trị thiếu.
  - ▶Giải pháp (Pandas): df.dropna().
  - ▶Cảnh báo: Đây là lựa chọn hạt nhân, có thể làm mất thông tin và gây thiên vị (bias).

- **4.11 Thay thế giá trị thiếu (Imputing):**
  - ▶Vấn đề: Dự đoán hoặc điền các giá trị bị thiếu.
  - ▶Giải pháp: Imputer(strategy='mean') (hoặc 'median', 'most_frequent').

4.10 & 4.11: Xử lý giá trị số bị thiếu

```python
# Load library
import numpy as np

# Create feature matrix
features = np.array([[1.1, 11.1],
                     [2.2, 22.2],
                     [3.3, 33.3],
                     [4.4, 44.4],
                     [np.nan, 55]])
```

```python
# Load data
dataframe = pd.DataFrame(features, columns=["feature_1", "feature_2"])

# Remove observations with missing values
dataframe.dropna()
```

```python
# Load library
from sklearn.preprocessing import Imputer

# Create imputer
mean_imputer = Imputer(strategy="mean", axis=0)

# Impute values

features_mean_imputed = mean_imputer.fit_transform(features)
```

```python
# Compare true and imputed values
print("True Value:", true_value)
print("Imputed Value:", features_mean_imputed[0,0])
```

```
True Value: 0.8730186114
Imputed Value: -3.05837272461
```

Xử lý dữ liệu định tính

Chuyển đổi các danh mục (nominal, ordinal) thành các giá trị số.

5.1 Mã hóa đặc trưng Nominal

- **Vấn đề:** Bạn có các lớp không có thứ tự (ví dụ: 'Texas', 'California').
- **Giải pháp:** One-Hot Encoding (OHE).
- OHE tạo ra một đặc trưng nhị phân mới cho mỗi lớp, tránh việc tạo ra một thứ tự giả.
- **Cách LÀM (scikit-learn):** LabelBinarizer().
- **Đa nhãn:** Sử dụng MultiLabelBinarizer() nếu một quan sát có thể thuộc nhiều lớp.

5.1 Mã hóa đặc trưng Nominal

```
# Import libraries
import numpy as np
from sklearn.preprocessing import LabelBinarizer, MultiLabelBinarizer

# Create feature
feature = np.array([["Texas"],
                    ["California"],
                    ["Texas"],
                    ["Delaware"],
                    ["Texas"]])

# Create one-hot encoder
one_hot = LabelBinarizer()

# One-hot encode feature
one_hot.fit_transform(feature)
```

```
array([[0, 0, 1],
       [1, 0, 0],
       [0, 0, 1],
       [0, 1, 0],
       [0, 0, 1]])
```

5.1 Mã hóa đặc trưng Nominal

```
# Create multiclass feature
multiclass_feature = [("Texas", "Florida"),
                      ("California", "Alabama"),
                      ("Texas", "Florida"),
                      ("Delware", "Florida"),
                      ("Texas", "Alabama")]

# Create multiclass one-hot encoder
one_hot_multiclass = MultiLabelBinarizer()

# One-hot encode multiclass feature
one_hot_multiclass.fit_transform(multiclass_feature)
```

```
array([[0, 0, 0, 1, 1],
       [1, 1, 0, 0, 0],
       [0, 0, 0, 1, 1],
       [0, 0, 1, 1, 0],
       [1, 0, 0, 0, 1]])
```

5.2 Mã hóa đặc trưng Ordinal

- **Vấn đề:** Bạn có các lớp có thứ tự tự nhiên (ví dụ: 'Low', 'Medium', 'High').
- **Giải pháp:** Ánh xạ (map) các lớp sang các giá trị số giữ nguyên thứ tự.
- **Ví dụ:**
  - scale_mapper = {"Low": 1, "Medium": 2, "High": 3}
  - dataframe["Score"].replace(scale_mapper)
- **Lưu ý:** Hãy cẩn thận về khoảng cách giữa các giá trị số.

5.2 Mã hóa đặc trưng Ordinal

```
dataframe = pd.DataFrame({"Score": ["Low",
                                    "Low",
                                    "Medium",
                                    "Medium",
                                    "High",
                                    "Barely More Than Medium"]})

scale_mapper = {"Low":1,
                "Medium":2,
                "Barely More Than Medium": 3,
                "High":4}

dataframe["Score"].replace(scale_mapper)
```

```
0    1
1    1
2    2
3    2
4    4
5    3
Name: Score, dtype: int64
```

5.2 Mã hóa đặc trưng Ordinal

```
scale_mapper = {"Low":1,
                "Medium":2,
                "Barely More Than Medium": 2.1,
                "High":3}

dataframe["Score"].replace(scale_mapper)
```

```
0    1.0
1    1.0
2    2.0
3    2.0
4    3.0
5    2.1
Name: Score, dtype: float64
```

5.3 Mã hóa Dictionary thành đặc trưng

- **Vấn đề:** Bạn có dữ liệu dạng dictionary (ví dụ: đếm từ) và muốn chuyển thành ma trận đặc trưng.
- **Giải pháp:** Sử dụng DictVectorizer của scikit-learn.
- **Ví dụ:**
  - data = [{"Red": 2, "Blue": 4}, {"Red": 4, "Blue": 3}]
  - dictvectorizer = DictVectorizer(sparse=False)
  - features = dictvectorizer.fit_transform(data)
- **Ứng dụng:** Rất phổ biến trong xử lý ngôn ngữ tự nhiên (NLP) cho các mô hình "bag of words".

5.4 & 5.5: Vấn đề dữ liệu định tính

- **5.4 Thay thế giá trị lớp bị thiếu:**
  - Giải pháp 1 (Tốt nhất): Sử dụng KNeighborsClassifier để dự đoán lớp bị thiếu.
  - Giải pháp 2 (Đơn giản): Imputer(strategy='most_frequent').
- **5.5 Xử lý lớp không cân bằng:**
  - Giải pháp 1 (Mô hình): class_weight="balanced".
  - Giải pháp 2 (Dữ liệu): Downsampling (giảm lớp đa số) hoặc Upsampling (tăng lớp thiểu số).