Module 2 - Tuần 3 - Tổng hợp kiến thức Buổi học số 5

Time-Series Team

Ngày 21 tháng 7 năm 2025

Buổi học số 3 (Thứ 5, 17/07/2025) bao gồm các nội dung chính:

- ***Phần 1: Firebase***
- ***Phần 2: Firebase + Python***
- ***Phần 3: Iris Dataset - Từ lưu trữ đến Machine Learning***

Phần 1: Firebase

1 Firebase là gì?

Firebase là một nền tảng Backend-as-a-Service (BaaS) do Google phát triển, cung cấp các dịch vụ đám mây giúp các nhà phát triển xây dựng ứng dụng dễ dàng, nhanh chóng và hiệu quả mà không cần phải tự xây dựng hạ tầng backend phức tạp. Firebase cung cấp nhiều dịch vụ tích hợp như:

- **Realtime Database và Cloud Firestore (NoSQL):** Hai hệ quản trị cơ sở dữ liệu NoSQL với khả năng lưu trữ dữ liệu phi cấu trúc, hỗ trợ cập nhật dữ liệu theo thời gian thực, giúp đồng bộ dữ liệu nhanh chóng giữa các client.
- **Authentication:** Hỗ trợ xác thực người dùng đa dạng thông qua email/password, Google, Facebook, Twitter, và nhiều phương thức khác.
- **Hosting:** Cung cấp dịch vụ lưu trữ và phân phối nội dung tĩnh như HTML, CSS, JavaScript với độ trễ thấp.
- **Cloud Functions:** Cho phép chạy mã serverless trên nền tảng đám mây, phản hồi các sự kiện phát sinh trong hệ thống.
- **ML Kit:** Bộ công cụ máy học giúp tích hợp các tính năng AI như nhận diện khuôn mặt, nhận dạng văn bản, dịch ngôn ngữ vào ứng dụng.

Mục tiêu chính của Firebase là giúp bạn tập trung vào phần logic nghiệp vụ, phân tích và xử lý dữ liệu, thay vì phải lo lắng về việc thiết lập và quản lý backend phức tạp, tiết kiệm thời gian và chi phí phát triển.

So sánh Firebase Realtime Database và Cloud Firestore

Firebase hiện có hai dịch vụ cơ sở dữ liệu NoSQL phổ biến: Firebase Realtime Database và Cloud Firestore. Cả hai đều hỗ trợ lưu trữ dữ liệu phi quan hệ, tuy nhiên có những điểm khác biệt quan trọng về cấu trúc dữ liệu, khả năng truy vấn và hiệu suất, phù hợp với các nhu cầu sử dụng khác nhau.

	Firebase Realtime Database	**Cloud Firestore**
Cấu trúc dữ liệu	Lưu trữ dữ liệu dưới dạng một cây JSON lớn, dữ liệu được tổ chức theo dạng nhánh (tree structure).	Lưu trữ theo mô hình document-collection tương tự MongoDB, dữ liệu được chia thành các document nhỏ gọn, tổ chức theo collection, dễ dàng quản lý và mở rộng.
Realtime và đồng bộ	Hỗ trợ đồng bộ dữ liệu theo thời gian thực rất tốt, phù hợp với các ứng dụng cần cập nhật liên tục.	Ngoài realtime, Firestore còn cung cấp khả năng truy vấn dữ liệu phức tạp mạnh mẽ hơn mà vẫn giữ được tính realtime.
Khả năng sử dụng	Dễ sử dụng cho các ứng dụng đơn giản, demo nhanh. Tuy nhiên khi mở rộng quy mô có thể gặp khó khăn do cấu trúc JSON lớn.	Thiết kế dễ dàng mở rộng quy mô trên nền Google Cloud, hỗ trợ tốt cho các ứng dụng có yêu cầu cao về truy vấn và quản lý dữ liệu phức tạp.
Khả năng truy vấn	Hạn chế trong truy vấn phức tạp, chủ yếu hỗ trợ truy vấn theo đường dẫn hoặc điều kiện đơn giản.	Hỗ trợ truy vấn với cú pháp mạnh mẽ như `WHERE`, truy vấn theo composite index, sắp xếp, phân trang, rất phù hợp với ứng dụng cần lọc và xử lý dữ liệu nâng cao.

Hạn chế khi sử dụng Firebase

Mặc dù Firebase là một nền tảng rất tiện lợi và mạnh mẽ, tuy nhiên nó cũng tồn tại một số hạn chế mà bạn cần lưu ý khi áp dụng trong các dự án Data Science hay ứng dụng thực tế:

- **Giới hạn về quy mô và chi phí:** Mặc dù Firebase có gói miễn phí, nhưng khi dữ liệu và lưu lượng truy cập tăng lên, chi phí có thể tăng rất nhanh, đặc biệt là với các ứng dụng realtime hoặc tải lớn.
- **Hạn chế về truy vấn phức tạp:** Cả Realtime Database và Firestore đều không hỗ trợ các truy vấn quá phức tạp như các hệ quản trị quan hệ (SQL) truyền thống, làm hạn chế khả năng phân tích và lọc dữ liệu nâng cao ngay trên database.
- **Giới hạn mô hình dữ liệu:** Firebase là NoSQL, nên không hỗ trợ tốt cho các mô hình dữ liệu có quan hệ phức tạp (ví dụ nhiều bảng liên kết) hoặc các nghiệp vụ cần join dữ liệu như trong các hệ thống quan hệ.
- **Phụ thuộc vào hệ sinh thái Google:** Khi sử dụng Firebase, bạn bị phụ thuộc vào hạ tầng và chính sách của Google, điều này có thể gây khó khăn khi muốn chuyển đổi hoặc tích hợp với các hệ thống khác.
- **Quản lý bảo mật phức tạp:** Việc thiết lập quy tắc bảo mật (Security Rules) cho Firebase không đơn giản, đòi hỏi hiểu rõ cách thức hoạt động để tránh lỗ hổng truy cập dữ liệu không mong muốn.
- **Giới hạn offline và đồng bộ:** Dù Firebase hỗ trợ offline cho client, nhưng khi ứng dụng phức tạp với nhiều thao tác đồng bộ, việc đảm bảo tính nhất quán dữ liệu vẫn có thể gặp khó khăn.

Hiểu rõ các hạn chế trên sẽ giúp bạn lựa chọn giải pháp phù hợp hơn cho từng bài toán, hoặc kết hợp Firebase với các công nghệ khác để tận dụng tối đa ưu điểm và giảm thiểu nhược điểm.

Phần 2: Firebase + Python

Firebase cung cấp nhiều dịch vụ backend dễ dùng và nhanh chóng để lưu trữ, truy vấn và đồng bộ dữ liệu. Để sử dụng Firebase trong dự án Data Science với Python, bạn cần làm theo các bước cơ bản sau đây:

Các bước thực hiện:

1. **Tạo Project Firebase và bật dịch vụ Realtime Database + Firestore:**
   - Truy cập vào https://console.firebase.google.com/ và tạo một Project mới.
   - Trong phần `Build` chọn bật dịch vụ `Realtime Database` và `Cloud Firestore` (Firestore thường được ưu tiên dùng cho ứng dụng phức tạp).
   - Chọn chế độ bảo mật (test hoặc production) phù hợp với mục đích sử dụng.
2. **Tạo Service Account và tải file JSON credentials:**
   - Vào phần `Project Settings` → `Service accounts`.
   - Tạo một tài khoản dịch vụ (service account) với quyền truy cập Firestore.
   - Tải file JSON chứa thông tin khóa xác thực, file này sẽ được dùng trong Python để kết nối.
3. **Cài đặt thư viện Python cần thiết:**

```
pip install firebase-admin
```

4. **Khởi tạo và kết nối Firebase trong Python:**
   - Sử dụng thư viện `firebase-admin` để kết nối và thao tác với Firestore.
   - Load file credentials JSON để xác thực.

Ví dụ truy cập và thao tác dữ liệu với Firestore:

```
from firebase_admin import credentials, firestore, initialize_app

# Initialize Firebase app with JSON key file
cred = credentials.Certificate("path/to/your_key.json")
initialize_app(cred)

# Get Firestore client
db = firestore.client()

# Point to the collection containing Iris flower data
collection = db.collection("iris_samples")

# Add a new document with ID "sample1"
collection.document("sample1").set({
    "sepal_length": 5.1,
    "sepal_width": 3.5,
    "petal_length": 1.4,
    "petal_width": 0.2,
    "species": "setosa"
})
```

Giải thích chi tiết

- `credentials.Certificate()` tạo một đối tượng thông tin xác thực từ tệp JSON, cho phép ứng dụng Python truy cập an toàn vào Firebase project.
- `initialize_app()` khởi tạo Firebase SDK với thông tin xác thực đã cung cấp.
- `firestore.client()` tạo một đối tượng client để thao tác với cơ sở dữ liệu Firestore.
- `db.collection("iris_samples")` trỏ đến bộ sưu tập có tên `iris_samples`, tương tự như một bảng (table) trong cơ sở dữ liệu quan hệ.
- `collection.document("sample1").set({...})` tạo mới hoặc cập nhật một tài liệu (document) có ID là `"sample1"` chứa các đặc trưng của mẫu hoa Iris.

Lưu ý khi làm việc với Firebase và Python

- Đảm bảo file JSON credentials được giữ bảo mật, không chia sẻ công khai hoặc commit lên các kho mã nguồn công khai.
- Kiểm tra kỹ quy tắc bảo mật Firestore để tránh dữ liệu bị truy cập trái phép.
- Firestore phù hợp để lưu trữ dữ liệu phi cấu trúc, dữ liệu dạng JSON, có thể mở rộng dễ dàng theo dạng document/collection.
- Việc thao tác với Firestore trong Python thường bất đồng bộ nhẹ, tuy nhiên nếu thao tác nhiều document lớn cần tối ưu hóa bằng batch write hoặc transaction.
- Để truy vấn và xử lý dữ liệu hiệu quả, nên kết hợp Firestore với các công cụ Data Science trong Python như pandas, scikit-learn.

Phần 3: Iris Dataset - Từ lưu trữ đến Machine Learning

1 Giới thiệu Dataset

Bộ dữ liệu Iris là một bộ dữ liệu kinh điển trong lĩnh vực Machine Learning với:

- 150 mẫu hoa Iris thuộc 3 loài: `setosa`, `versicolor`, và `virginica`.
- Mỗi mẫu có 4 đặc trưng hình thái chính: chiều dài và chiều rộng của sepal (cánh hoa ngoài) và petal (cánh hoa trong).

Dữ liệu thường được dùng để bài toán phân loại đa lớp (multi-class classification).

2 Tải dữ liệu và upload lên Firebase Firestore

Để làm việc với dataset, ta có thể dùng thư viện `sklearn` để tải dữ liệu, chuyển thành `pandas DataFrame` và lần lượt upload từng mẫu lên Firestore.

```
from sklearn.datasets import load_iris
import pandas as pd

# Load the Iris dataset
data = load_iris()

# Convert the data to a DataFrame with clear column names
df = pd.DataFrame(data.data, columns=data.feature_names)

# Add a new column for species names as text
df['species'] = [data.target_names[i] for i in data.target]

# Upload each data sample to Firestore
for i, row in df.iterrows():
    collection.document(f"sample_{i}").set(row.to_dict())
```

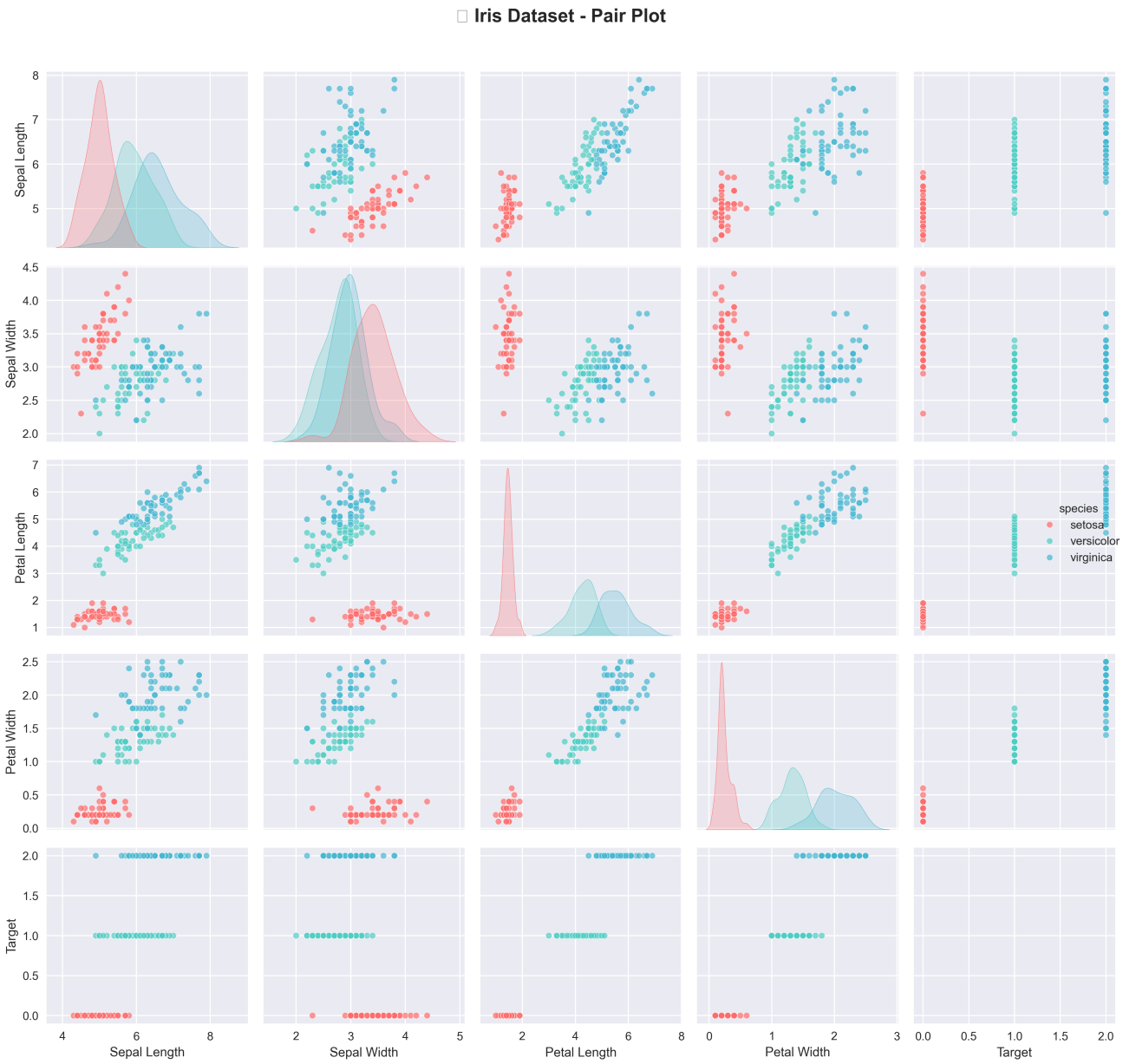
Giải thích: mỗi dòng dữ liệu sẽ được chuyển thành dictionary và lưu trong một document riêng biệt trong collection `iris_samples`. Đây là cách lưu dữ liệu phi cấu trúc rất hiệu quả với Firestore.

3 Visualize dữ liệu

Trước khi huấn luyện mô hình, bạn có thể trực quan hóa dữ liệu để hiểu rõ phân bố đặc trưng giữa các loài hoa.

- Sử dụng biểu đồ **scatter plot** để biểu diễn mối quan hệ giữa `petal length` và `petal width`.
- Phân biệt các loài hoa bằng màu sắc khác nhau để dễ nhận biết.

Điều này giúp phát hiện các mẫu phân biệt rõ ràng và các điểm có thể gây nhầm lẫn.



Hình 1: Iris Dataset - Pair Plot

4 Huấn luyện mô hình Machine Learning

Sau khi chuẩn bị dữ liệu, ta có thể xây dựng các mô hình phân loại để dự đoán loài hoa dựa trên các đặc trưng.

- Các mô hình thường dùng: `Decision Tree`, `Random Forest`, `Support Vector Machine (SVM)`.
- Chia dữ liệu thành tập huấn luyện và kiểm thử (train/test split).
- Đánh giá mô hình bằng các chỉ số như accuracy.

Ví dụ huấn luyện `Random Forest`:

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Prepare features and labels
X = df[data.feature_names]
y = df['species']

# Split into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Evaluate accuracy on the test set
accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2f}")
```

5 Xây dựng giao diện dự đoán bằng Streamlit

Streamlit là thư viện Python giúp tạo giao diện web tương tác rất dễ dàng, phù hợp để demo nhanh mô hình ML.

Các bước chính

1. Tạo giao diện nhập liệu cho 4 đặc trưng của hoa Iris: sepal length, sepal width, petal length, petal width.
2. Khi người dùng nhập dữ liệu và nhấn nút dự đoán, gọi mô hình ML đã huấn luyện để dự đoán loài hoa.
3. Hiển thị kết quả dự đoán cùng với độ tin cậy (confidence).
4. Gửi kết quả dự đoán và thông tin đầu vào lên Firebase Firestore để lưu trữ.

Ví dụ mã nguồn Streamlit đơn giản:

Simple Streamlit Source Code Example:

```python
import streamlit as st
import joblib

# Load the pre-trained model (saved as a .pkl file)
model = joblib.load("rf_model.pkl")

st.title("Iris Flower Species Prediction")

# Enter feature values
sepal_length = st.number_input("Sepal Length (cm)", min_value=0.0, max_value=10.0, value
    =5.0, step=0.1)
sepal_width = st.number_input("Sepal Width (cm)", min_value=0.0, max_value=10.0, value=3.5,
     step=0.1)
petal_length = st.number_input("Petal Length (cm)", min_value=0.0, max_value=10.0, value
    =1.4, step=0.1)
```

```
petal_width = st.number_input("Petal Width (cm)", min_value=0.0, max_value=10.0, value=0.2,
    step=0.1)

if st.button("Predict Flower Species"):
    features = [[sepal_length, sepal_width, petal_length, petal_width]]
    prediction = model.predict(features)[0]
    prediction_proba = model.predict_proba(features).max()

    st.success(f"Prediction: {prediction}")
    st.info(f"Confidence: {prediction_proba:.2%}")

    # TODO: Send the prediction result to Firebase Firestore for storage
```

Triển khai lên cloud

Bạn có thể sử dụng `secrets.toml` để bảo mật khóa Firebase trong dự án Streamlit Cloud và deploy app trực tiếp lên Streamlit Cloud với vài bước đơn giản.

6 Ứng dụng thực tế cho Data Scientist

Tình huống sử dụng Firebase

- Thu thập dữ liệu cảm biến thời gian thực (IoT) gửi lên Firebase để xử lý và phân tích.
- Ghi lại hành động của người dùng hoặc feedback để thực hiện A/B testing.
- Tích hợp với mobile app để gửi dữ liệu dự đoán hoặc cập nhật mô hình AI.

ML Pipeline đơn giản hóa với Firebase

Firebase có thể đóng vai trò như một **Data Lake** NoSQL nhỏ gọn, thay vì cần hạ tầng lớn như BigQuery hay Amazon S3, bạn có thể:

- Thu thập dữ liệu trực tiếp
- Lưu trữ dữ liệu dạng phi cấu trúc
- Huấn luyện mô hình ML với dữ liệu realtime
- Triển khai dự đoán nhanh, realtime
- Truy vết và lưu lại kết quả dự đoán