

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT

BÁO CÁO HỌC THUẬT

Tên báo cáo:

**Tác động của lựa chọn biến đầu vào đến hiệu quả dự báo đường cong ĐVLGK
bằng các thuật toán máy học**

**GV thực hiện: ThS. Bùi Thị Ngân
Bộ môn: Địa chất dầu khí**

Hà Nội, 01/2026

MỞ ĐẦU

Log DT (Sonic) là tham số quan trọng trong: Mô hình vận tốc, Phân tích địa tầng, Đánh giá địa cơ học. Tuy nhiên, trong thực tế: DT thường thiếu, nhiều hoặc không được đo ở nhiều giếng. Do đó, dự báo DT từ các đường log khác là một bài toán thực tiễn và cần thiết.

Các phương pháp học máy, đặc biệt là Random Forest, cho thấy khả năng: Mô hình hóa quan hệ phi tuyến, Xử lý dữ liệu log phức tạp. Tuy nhiên, phần lớn các nghiên cứu thường: Tập trung vào độ chính xác dự báo và Ít phân tích vai trò của từng biến đầu vào.

Các log địa vật lý thường: Có tương quan mạnh với nhau, Mang thông tin chồng chéo. Do đó, việc lựa chọn biến đầu vào có thể làm thay đổi đáng kể hành vi của mô hình và ảnh hưởng đến khả năng tổng quát hóa sang giếng mới.

Vì vậy, đánh giá tầm ảnh hưởng của lựa chọn biến đầu vào là bước không thể bỏ qua.

1. Gọi ra các thư viện cần sử dụng

Thư viện lasio: Đọc và làm việc với file .LAS – định dạng chuẩn trong địa vật lý giếng khoan.

Thư viện Pandas - Xử lý dữ liệu dạng bảng (DataFrame)

Thư viện Numpy: Xử lý tính toán số học và mảng số, ví dụ chuẩn hóa log

Thư viện matplotlib.pyplot - Vẽ biểu đồ cơ bản, đặc biệt là log theo chiều sâu, ví dụ plot DT theo depth, So sánh DT thật vs DT dự báo

Thư viện seaborn - Vẽ biểu đồ thống kê, đẹp hơn matplotlib để hiểu quan hệ giữa logs, ví dụ biểu đồ heatmap thể hiện tương quan giữa các logs

Import thuật toán Random Forest cho bài toán hồi quy (predict giá trị liên tục). Mỗi cây quyết định $\approx$ một cách “diễn giải địa chất” khác nhau Forest = tổng hợp nhiều cách diễn giải $\rightarrow$ dự báo ổn định hơn

`train_test_split`

Chia dữ liệu thành:

- Tập huấn luyện (train)
- Tập kiểm tra (test)
- chia ngẫu nhiên theo độ sâu, không theo giếng, tức là:
 - Train và test có thể cùng giếng
 - Nhưng khác điểm đo (depth)

`GridSearchCV`

Tự động tìm bộ hyperparameter tốt nhất cho Random Forest.

`StandardScaler`

Nếu chỉ áp dụng thuật toán RF thì không bắt buộc chuẩn hóa các giá trị các đường log vì RF dựa trên so sánh ngưỡng và thứ tự giá trị chứ không dựa trên khoảng cách. Tuy nhiên để mở rộng là workflow chung cho nhiều model thì vẫn có bước này.

mean_absolute_error, mean_squared_error: Đánh giá độ chính xác của DT dự báo (Sai số trung bình của DT ($\mu\text{s}/\text{ft}$) ít nhạy với outlier còn MSE phạt mạnh sai số lớn)

```
# 1. Import thư viện
import lasio
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

Trong bài toán dự báo log DT, các giá trị outlier trong logs địa vật lý thường không mang ý nghĩa địa chất mà xuất phát từ lỗi đo hoặc điều kiện lỗ khoan xấu. Do đó, thay vì loại bỏ các điểm đo này, phương pháp Winsorizing dựa trên IQR được sử dụng để giới hạn biên giá trị, giúp mô hình Random Forest học được xu hướng vật lý chính mà vẫn giữ nguyên cấu trúc dữ liệu theo chiều sâu.

def – Khai báo hàm,

- series = log GR / RHOB / NPHI / DT của một giếng
- Hàm được áp dụng riêng cho từng log
- k: hệ số mở rộng IQR (mặc định 1.5 – chuẩn thống kê)

Kiểm tra toàn bộ giá trị là NaN – khi log không tồn tại ở một giếng, ví dụ đường NPHI, hoặc không được đo ở một khoảng, ví dụ DT, sẽ gây lỗi khi tính tứ phân vị.

Tính các phân vị (Quartiles)

Q1: giá trị tại 25% dữ liệu

Q3: giá trị tại 75% dữ liệu

Đại diện cho vùng giá trị “bình thường” của log

Tính IQR (Interquartile Range) - Độ rộng của 50% dữ liệu trung tâm

Winsorizing – cắt biên thay vì xóa dữ liệu

- Giá trị < lower → gán = lower
- Giá trị > upper → gán = upper

Hàm loại cắt biên outliers bằng IQR và Winsorizing

```
def remove_outliers_iqr(series, k=1.5):  
    if series.isnull().all():  
        return series  
    Q1 = series.quantile(0.25)  
    Q3 = series.quantile(0.75)  
    IQR = Q3 - Q1  
    lower = Q1 - k * IQR  
    upper = Q3 + k * IQR  
    return series.clip(lower, upper)
```

2. Đọc và xử lý dữ liệu từ các file .las

```
def read_las(well_name, filepath):  
    las = lasio.read(filepath)  
    df = las.df().reset_index()  
    df['WELL'] = well_name
```

3. Đọc 4 giếng từ file .las với đường dẫn đầy đủ

```
demo1 = read_las('Demo1', 'D:/AI/TGT/DT/Demo1.las')  
demo2 = read_las('Demo2', 'D:/AI/TGT/DT/Demo2.las')  
demo3 = read_las('Demo3', 'D:/AI/TGT/DT/Demo3.las')  
demo5 = read_las('Demo5', 'D:/AI/TGT/DT/Demo5.las')
```

Áp dụng cho tất cả input logs

```
input_logs = ['GR', 'RHOB', 'LLD_LOG', 'NPHI']  
for col in input_logs:  
    if col in df.columns:  
        df[col] = remove_outliers_iqr(df[col])  
return df[['WELL', 'DEPTH', 'GR', 'RHOB', 'LLD_LOG', 'NPHI', 'DT']]
```

Trong tập dữ liệu huấn luyện, các điểm đo bị thiếu bất kỳ log nào đều được loại bỏ nhằm đảm bảo mô hình Random Forest chỉ học trên các mẫu có đầy đủ thông tin đầu vào và giá trị DT thực. Ngược lại, đối với giếng ứng dụng không có DT, dữ liệu chỉ được lọc theo các logs đầu vào, phản ánh đúng bài toán dự báo DT trong thực tế khai thác.

Chuẩn bị hai tập dữ liệu khác nhau với hai mục đích khác nhau:

df_train → dùng để huấn luyện & đánh giá mô hình Random Forest

demo5_clean → dùng để dự báo DT cho giếng không có DT (giếng ứng dụng)

demo1, demo2, demo3 tương ứng 3 giếng có đầy đủ logs + DT, là “ground truth” cho bài toán học máy

.dropna() - Loại bỏ mọi depth có ít nhất 1 log bị thiếu bởi vì:

- Random Forest không xử lý được NaN
- DT là target → không thể train nếu thiếu DT

Ví dụ:

- Một depth thiếu RHOB → không có thông tin mật độ
- Một depth thiếu DT → không có nhãn để học

Depth đó phải bị loại khỏi tập train

Làm sạch dữ liệu giếng ứng dụng (Demo5) - khác hoàn toàn so với tập train, vì Demo5 không có DT → mục tiêu là dự báo DT, không phải huấn luyện

- Chỉ yêu cầu đầy đủ các log đầu vào
- Không quan tâm DT (vì DT không tồn tại)

demo5_clean - Kiểm tra nhanh:

- Còn NaN hay không
- Số lượng mẫu
- Logs đầu vào đã đủ chưa

```
# 4. Làm sạch dữ liệu
df_train = pd.concat([
    demo1.dropna(),
    demo2.dropna(),
    demo3.dropna()
], ignore_index=True)

# Đối với Demo5, chỉ bỏ NaN ở input
demo5_clean = demo5.dropna(subset=['GR', 'RHOB', 'LLD_LOG', 'NPHI']).copy()

demo5_clean
```

	WELL	DEPTH	GR	RHOB	LLD_LOG	NPHI	DT
1361	Demo5	3366.0656	51.3712	2.1424	1.141838	0.3275	93.9110
1362	Demo5	3366.1656	51.5931	2.1424	1.141838	0.3275	94.5115
1364	Demo5	3366.3656	48.7850	2.1424	1.141838	0.3275	94.9604
1365	Demo5	3366.4656	48.5556	2.1424	1.141838	0.3275	95.5234
1367	Demo5	3366.6656	48.0723	2.1424	1.141838	0.3275	95.8275
...	...	...	...	...	...	...	...
8205	Demo5	4050.4656	90.3310	2.4146	0.781015	0.1210	NaN
8207	Demo5	4050.6656	87.9812	2.4142	0.804644	0.1199	NaN
8208	Demo5	4050.7656	80.8527	2.4137	0.825361	0.1182	NaN
8210	Demo5	4050.9656	84.1000	2.4157	0.845538	0.1213	NaN
8211	Demo5	4051.0656	88.8943	2.4230	0.847579	0.1273	NaN

4496 rows × 7 columns

Mặc dù Random Forest không yêu cầu bắt buộc phải chuẩn hóa dữ liệu, StandardScaler vẫn được sử dụng trong bài này nhằm đảm bảo quy trình Machine Learning nhất quán và dễ mở rộng sang các thuật toán khác. Bộ chuẩn hóa được

huấn luyện chỉ trên tập dữ liệu huấn luyện và sau đó áp dụng cho giềng ứng dụng, nhằm tránh hiện tượng rò rỉ thông tin (data leakage).

Tách biến đầu vào (X) và biến mục tiêu (y)

- X: các logs đầu vào dùng để dự báo DT
- y: log DT thực đo (ground truth)

scaler = StandardScaler() - Khởi tạo bộ chuẩn hóa dữ liệu

Chuẩn hóa mỗi feature về:

- Mean = 0
- Standard deviation = 1

Công thức:

$$X_{\text{scaled}} = (X - \mu) / \sigma$$

- fit_transform

fit():

- Tính mean & std của từng log
- Chỉ dựa trên df_train

transform():

- Áp dụng chuẩn hóa theo các giá trị đã học

Không được dùng dữ liệu Demo5 để fit scaler → tránh data leakage

Các log: GR, RHOB, NPHI, LLD_LOG có:

- Đơn vị khác nhau
- Biên độ khác nhau

Chuẩn hóa giúp:

- Dữ liệu dễ hiểu
- Dễ so sánh
- Dễ mở rộng sang model khác

- Tránh feature có biên độ lớn “áp đảo” trong GridSearch

Áp dụng scaler cho giếng Demo5 - Dùng cùng một scaler đã học từ dữ liệu train

```
# 5. Chuẩn hóa dữ liệu
X = df_train[['GR', 'RHOB', 'LLD_LOG', 'NPHI']]
y = df_train['DT']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Áp dụng scaler đã huấn luyện vào dữ liệu Demo5
X_demo5 = demo5_clean[['GR', 'RHOB', 'LLD_LOG', 'NPHI']]
X_demo5_scaled = scaler.transform(X_demo5)
X_demo5_scaled
X_scaled

array([[ -1.70773536, -2.03202549, -2.51985614,  2.47219781],
       [ -1.75459411, -2.0112617 , -2.51985614,  2.47219781],
       [ -1.83065818, -2.04240739, -2.51985614,  2.47219781],
       ...,
       [  2.3893791 ,  0.69086277, -0.12820192, -0.92074464],
       [  2.46046263,  0.53513433, -0.14553979, -0.90529584],
       [  2.51628423,  0.32844021, -0.08226571, -0.88212264]])
```

Tách dữ liệu đã chuẩn hóa thành:

- Tập huấn luyện (train): để Random Forest học quan hệ log → DT
- Tập kiểm tra (test): để đánh giá mô hình trên dữ liệu *chưa thấy*

Đây là bước bắt buộc trong mọi bài toán ML có giám sát.

test_size=0.2

- 20% dữ liệu → tập test
- 80% dữ liệu → tập train

Đây là tỷ lệ phổ biến, cân bằng giữa:

- Dữ liệu đủ để học
- Dữ liệu đủ để kiểm tra

random_state=42

- Cố định seed cho bộ sinh số ngẫu nhiên
- Đảm bảo:
 - Kết quả có thể lặp lại
 - So sánh các mô hình là công bằng

Con số 42:

- Không có ý nghĩa địa chất
- Chỉ là “con số truyền thống” trong cộng đồng ML

Trong nghiên cứu này, dữ liệu được chia ngẫu nhiên theo các điểm đo độ sâu nhằm đánh giá khả năng mô hình học được mối quan hệ giữa các logs địa vật lý và log DT. Cách chia này phù hợp cho mục đích minh họa thuật toán và quy trình Machine Learning cơ bản, tuy nhiên chưa phản ánh đầy đủ khả năng tổng quát hóa của mô hình đối với các giếng hoàn toàn mới.

```
# 4. Tách train/test
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42)
```

Khai báo không gian tìm kiếm hyperparameter

Tổ hợp các tham số được tạo thủ công, sau đó: GridSearch sẽ:

- Thử tất cả các tổ hợp
- Chọn mô hình cho DT ổn định nhất

n_estimators = 150

- Số lượng cây quyết định trong rừng

Với logs giếng khoan, tăng >200 cây thường không cải thiện nhiều hiệu suất mô hình và mất thời gian tính toán.

max_depth

- Giới hạn độ sâu của mỗi cây
- Kiểm soát độ phức tạp mô hình

`min_samples_split` - Số mẫu tối thiểu tại một node để node đó được phép tiếp tục chia, nếu Giá trị lớn → cây ít chia hơn

`min_samples_leaf`: Số mẫu tối thiểu bắt buộc phải có trong mỗi lá (leaf), là tham số ảnh hưởng trực tiếp đến hình dạng log DT.

Cả hai đều nhằm giảm độ phức tạp của cây, nhưng ở hai giai đoạn khác nhau.

`rf = RandomForestRegressor(random_state=42)` - Khởi tạo Random Forest

`random_state=42`

- Cố định tính ngẫu nhiên:
 - Bootstrap samples
 - Feature selection

Đảm bảo kết quả:

- Có thể tái lập
- So sánh được giữa các lần chạy

```
#5. Huấn Luyện mô hình + GridSearchCV
param_grid = {
    'n_estimators': [ 150],
    'max_depth': [10, 12, 16, 20],
    'min_samples_split': [2,5,10],
    'min_samples_leaf': [2,5]
}

rf = RandomForestRegressor(random_state=42)

grid_search = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1, scoring='neg_mean_absolute_error')
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)

Best Parameters: {'max_depth': 16, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 150}
```

Đo lường mức độ mô hình dự báo DT gần với DT thực đến đâu trên tập test chưa dùng để huấn luyện.

- Dùng mô hình RF tốt nhất (sau GridSearch)

- Dự báo DT cho các sample chưa thấy trong quá trình train

```
# Bước 6: Đánh giá mô hình
```

```
y_pred = best_model.predict(X_test)
from sklearn.metrics import mean_squared_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

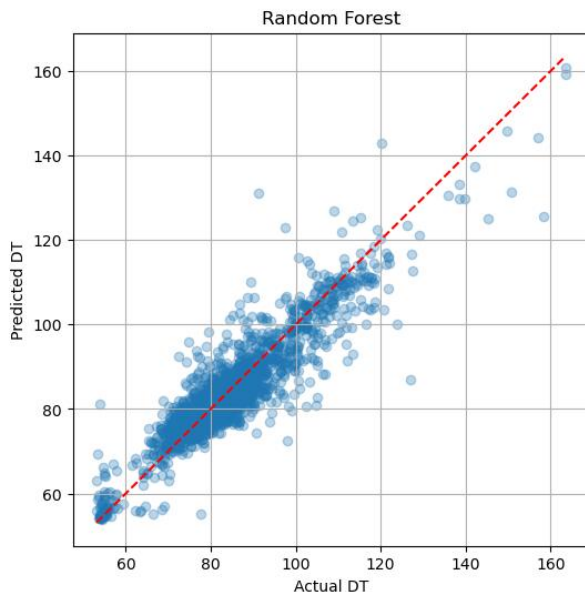
MAE: 2.76

RMSE: 4.21

R²: 0.84

Mô hình Random Forest cho kết quả MAE và RMSE ở mức chấp nhận được đối với log DT, cho thấy mô hình học được mối quan hệ phi tuyến giữa các logs địa vật lý.

Sau khi huấn luyện mô hình Random Forest trên dữ liệu của ba giếng khoan, mô hình được sử dụng để dự báo log DT cho giếng Demo5 – giếng không tham gia vào quá trình huấn luyện.



Những log nào (GR, RHOB, LLD_LOG, NPHI) ảnh hưởng mạnh nhất đến DT trong mô hình Random Forest?

Khai báo danh sách tên biến

Sắp xếp theo độ quan trọng giảm dần

Một biến giúp giảm sai số trong cây bao nhiêu, Tổng các importance = 1

Kết quả Feature Importance cho thấy NPHI là log chi phối mạnh nhất trong quá trình dự báo DT, phản ánh quan hệ vật lý của độ rỗng đối với vận tốc sóng âm. GR đứng thứ hai, cho thấy mô hình Random Forest đã học được ảnh hưởng gián tiếp của lithology đến DT. Mô hình này học rất tốt về mặt thống kê.

```
import matplotlib.pyplot as plt
import numpy as np

# Danh sách tên các biến (sau feature engineering)
feature_names = ['GR', 'RHOB', 'LLD_LOG', 'NPHI']

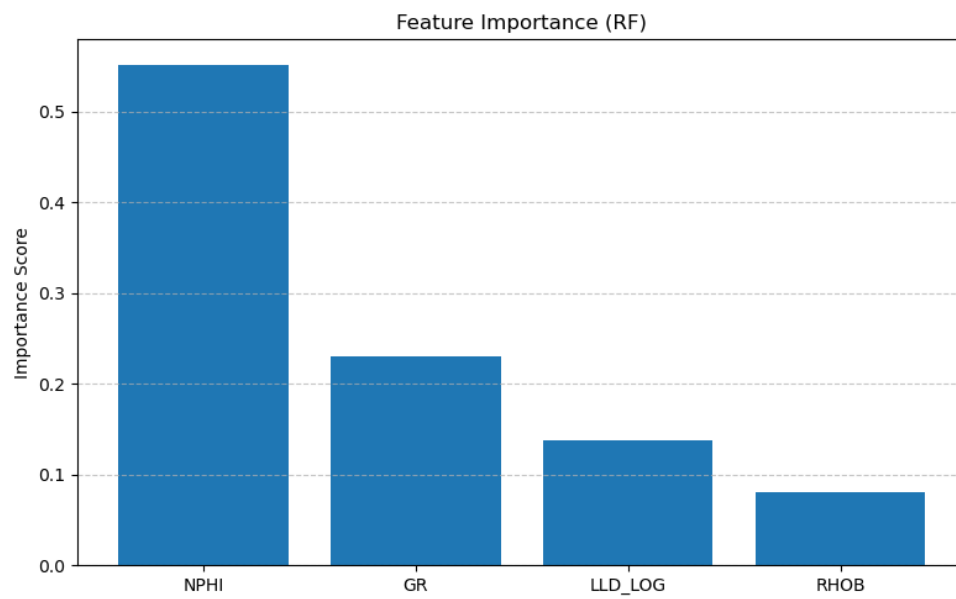
importances = final_model.feature_importances_

# Sắp xếp giảm dần
indices = np.argsort(importances)[::-1]

# Tạo DataFrame hiển thị tên biến và độ quan trọng
importance_df = pd.DataFrame({
    'Feature': [feature_names[i] for i in indices],
    'Importance': importances[indices]
})

# In ra bảng giá trị Feature Importance
print("=== Feature Importance ===")
print(importance_df.to_string(index=False))
```

```
=== Feature Importance ===
Feature  Importance
    NPHI      0.551683
     GR      0.230078
LLD_LOG      0.137899
    RHOB      0.080339
```



KẾT LUẬN

Kết quả Feature Importance cho thấy NPHI đóng vai trò chi phối trong dự báo DT và log GR góp phần quan trọng thông qua ảnh hưởng gián tiếp của lithology.

Mô hình có thể dự báo tốt nhưng vẫn tiềm ẩn rủi ro tổng quát hóa vì việc mô hình dựa mạnh vào một số biến nhất định (ví dụ NPHI, GR) giúp giảm sai số trong tập dữ liệu huấn luyện nhưng đồng thời có thể làm tăng rủi ro khi áp dụng cho giếng mới, đặc biệt trong trường hợp thay đổi facies hoặc môi trường trầm tích.