🏠 BÁO CÁO KỸ THUẬT: DỰ ĐOÁN GIÁ NHÀ

VietAI - Foundations of Machine Learning

Final Project Report

Ngày hoàn thành: Tháng 12/2025

Công nghệ sử dụng: Python, Scikit-learn, PyTorch, FastAPI, Streamlit

---

📑 MỤC LỤC

---

1. GIỚI THIỆU BÀI TOÁN

1.1 Tổng quan

Dự án này xây dựng một hệ thống Machine Learning hoàn chỉnh để **dự đoán giá nhà** dựa trên các đặc điểm của ngôi nhà. Hệ thống bao gồm từ việc xử lý dữ liệu thô, huấn luyện mô hình, cho đến triển khai sản phẩm để người dùng cuối có thể tương tác.

1.2 Dataset

- **Nguồn:** Kaggle - House Prices: Advanced Regression Techniques
- **URL:** https://www.kaggle.com/c/house-prices-advanced-regression-techniques
- **Kích thước:**
  - Training set: 1,460 samples
  - Test set: 1,459 samples
  - Features: 79 variables (36 numerical, 43 categorical)
- **Target variable:** `SalePrice` - Giá bán nhà (USD)

1.3 Mục tiêu dự án

Mục tiêu	Mô tả
Data Validation	Kiểm định chất lượng dữ liệu đầu vào
Feature Engineering	Tạo đặc trưng mới để tối ưu hiệu năng
Model Training	Huấn luyện và so sánh nhiều mô hình
API Deployment	Triển khai mô hình dưới dạng REST API
Web Interface	Xây dựng giao diện web tương tác
Deep Learning	Thử nghiệm mạng nơ-ron với PyTorch

1.4 Cấu trúc dự án

```
Final project/
├── data/
│   ├── raw/              # Dữ liệu gốc
│   └── processed/        # Dữ liệu đã xử lý
├── notebooks/
│   ├── 01_EDA.ipynb      # Exploratory Data Analysis
│   └── 02_Training.ipynb # Model Training Pipeline
├── src/
│   ├── config.py         # Cấu hình
│   ├── data_validation.py# Kiểm định dữ liệu
│   ├── preprocessing.py  # Tiền xử lý
│   ├── model_training.py # Huấn luyện mô hình
│   └── deep_learning.py  # Neural Network (PyTorch)
├── api/
│   └── main.py           # FastAPI Backend
├── app/
│   └── streamlit_app.py  # Streamlit Frontend
├── models/               # Mô hình đã lưu
├── reports/              # Báo cáo và biểu đồ
└── requirements.txt      # Dependencies
```

---

2. PHÂN TÍCH VÀ KHÁM PHÁ DỮ LIỆU

2.1 Thống kê mô tả

Target variable (SalePrice)

Thống kê	Giá trị
Mean	$180,921
Median	$163,000
Std	$79,443
Min	$34,900
Max	$755,000
Skewness	1.88 (Right-skewed)

Nhận xét: Phân phối giá nhà lệch phải, cần áp dụng **log transformation** để chuẩn hóa.

Các biến số quan trọng

Feature	Mean	Std	Min	Max
GrLivArea	1,515 sq ft	525	334	5,642
OverallQual	6.1	1.4	1	10
YearBuilt	1971	30	1872	2010
TotalBsmtSF	1,057 sq ft	439	0	6,110
GarageCars	1.8	0.7	0	4

2.2 Phân tích giá trị thiếu (Missing Values)

📊 Columns with Missing Values: 19/81

Top Missing Columns:

Column	Missing	%
PoolQC	1,453	99.5%
MiscFeature	1,406	96.3%
Alley	1,369	93.8%
Fence	1,179	80.8%
MasVnrType	872	59.7%
FireplaceQu	690	47.3%
LotFrontage	259	17.7%
GarageType	81	5.5%
BsmtQual	37	2.5%
BsmtCond	37	2.5%

Chiến lược xử lý:

- **PoolQC, Alley, Fence, MiscFeature:** NA có nghĩa "không có" → Fill với "None"
- **LotFrontage:** Fill với median theo Neighborhood
- **Garage, Basement features:** NA nghĩa "không có" → Fill với 0 hoặc "None"

2.3 Phân tích tương quan

Top 10 Features tương quan cao nhất với SalePrice

Rank	Feature	Correlation
1	OverallQual	0.79
2	GrLivArea	0.71
3	GarageCars	0.64
4	GarageArea	0.62
5	TotalBsmtSF	0.61
6	1stFlrSF	0.61
7	FullBath	0.56
8	TotRmsAbvGrd	0.53
9	YearBuilt	0.52
10	YearRemodAdd	0.51

Nhận xét:

- `OverallQual` và `GrLivArea` có tương quan mạnh nhất

- Các features liên quan đến diện tích và chất lượng là quan trọng nhất

2.4 Phân tích Outliers

```
Outliers detected (IQR method):
- GrLivArea: 31 outliers
- SalePrice: 61 outliers
- LotArea: 113 outliers

Critical outliers to remove:
- 2 houses with GrLivArea > 4000 sq ft and SalePrice < $300,000
  (Unusual pattern - likely data entry errors)
```

2.5 Biểu đồ phân tích

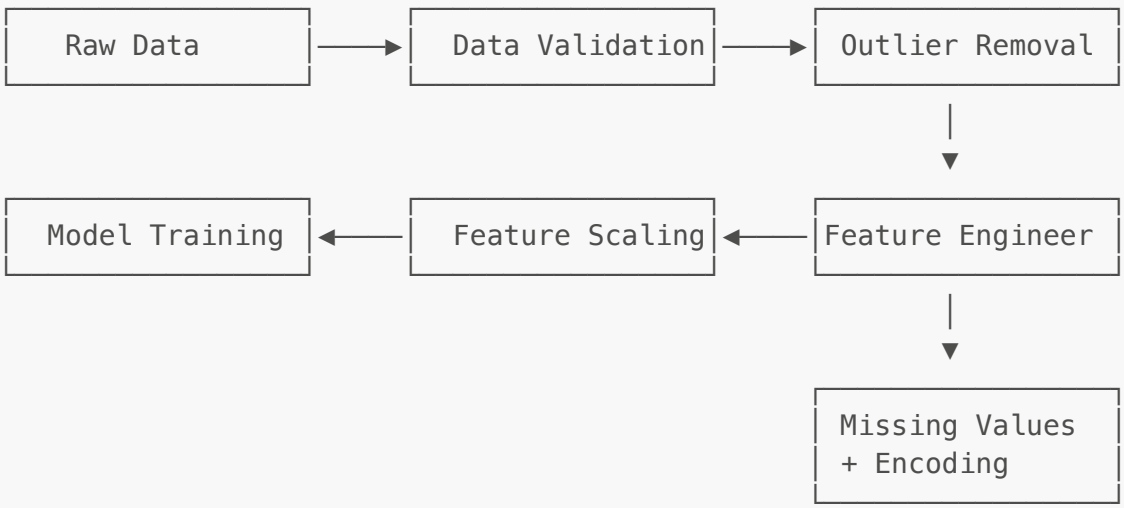
(Các biểu đồ được lưu trong thư mục `reports/`)

1. **missing_values.png** - Heatmap giá trị thiếu
2. **target_distribution.png** - Phân phối SalePrice (original vs log)
3. **correlation_heatmap.png** - Ma trận tương quan top 15 features
4. **scatter_plots.png** - Scatter plots các features quan trọng
5. **categorical_analysis.png** - Box plots biến phân loại

---

3. PIPELINE XỬ LÝ DỮ LIỆU

3.1 Tổng quan Pipeline



3.2 Data Validation (Kiểm định dữ liệu)

Schema Definition:

```
# Định nghĩa schema cho các cột quan trọng
schema = {
    'OverallQual': {'type': 'numeric', 'min': 1, 'max': 10, 'nullable':
False},
    'GrLivArea': {'type': 'numeric', 'min': 0, 'nullable': False},
    'YearBuilt': {'type': 'numeric', 'min': 1800, 'max': 2025, 'nullable':
False},
    'Neighborhood': {'type': 'categorical', 'nullable': False},
    'ExterQual': {'type': 'categorical', 'values': ['Ex', 'Gd', 'TA',
'Fa', 'Po']},
    ...
}
```

Data Quality Checks:

- ✅ Kiểm tra kiểu dữ liệu
- ✅ Kiểm tra giá trị null
- ✅ Kiểm tra khoảng giá trị hợp lệ
- ✅ Kiểm tra giá trị phân loại hợp lệ

3.3 Outlier Removal

```
# Loại bỏ outliers đặc biệt (identified in EDA)
outlier_mask = (df['GrLivArea'] > 4000) & (df['SalePrice'] < 300000)
df_clean = df[~outlier_mask]

# Kết quả: Loại bỏ 2 outliers
# Training data: 1,460 → 1,458 rows
```

3.4 Feature Engineering

Các features mới được tạo (14 features):

Feature	Công thức	Mô tả
TotalSF	TotalBsmtSF + 1stFlrSF + 2ndFlrSF	Tổng diện tích
TotalBath	FullBath + 0.5×Half + BsmtFull...	Tổng số phòng tắm
TotalPorchSF	WoodDeck + OpenPorch + Enclosed...	Tổng diện tích hiên
HouseAge	YrSold - YearBuilt	Tuổi nhà
RemodAge	YrSold - YearRemodAdd	Thời gian từ cải tạo
WasRemodeled	YearRemodAdd ≠ YearBuilt ? 1 : 0	Đã cải tạo?
HasPool	PoolArea > 0 ? 1 : 0	Có hồ bơi?
HasGarage	GarageArea > 0 ? 1 : 0	Có garage?
HasFireplace	Fireplaces > 0 ? 1 : 0	Có lò sưởi?
HasBasement	TotalBsmtSF > 0 ? 1 : 0	Có tầng hầm?
Has2ndFloor	2ndFlrSF > 0 ? 1 : 0	Có tầng 2?
QualityArea	OverallQual × GrLivArea	Chất lượng × DT
OverallScore	OverallQual × OverallCond	Điểm tổng thể
GarageEfficiency	GarageCars / (GarageArea + 1)	Hiệu suất garage

3.5 Missing Value Handling

```
# Chiến lược xử lý giá trị thiếu

# 1. Columns có NA nghĩa "không có feature"
NONE_FILL_COLS = ['Alley', 'PoolQC', 'Fence', 'MiscFeature',
'FireplaceQu',
                  'GarageType', 'GarageFinish', 'GarageQual',
'GarageCond',
                  'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
'BsmtFinType2']
→ Fill với 'None'

# 2. Numerical columns
→ Fill với median

# 3. Categorical columns khác
→ Fill với mode
```

Kết quả: Missing values: 7,829 → 0

3.6 Categorical Encoding

Quality Mapping (Ordinal Encoding):

```
QUALITY_MAPPING = {
    'Ex': 5,   # Excellent
    'Gd': 4,   # Good
    'TA': 3,   # Typical/Average
    'Fa': 2,   # Fair
    'Po': 1,   # Poor
    'None': 0  # No feature
}

# Áp dụng cho: ExterQual, ExterCond, BsmtQual, BsmtCond,
#              HeatingQC, KitchenQual, FireplaceQu, GarageQual,
GarageCond, PoolQC
```

One-Hot Encoding:

```
# Các biến phân loại còn lại
X = pd.get_dummies(X, drop_first=True)

# Kết quả: 79 features → 241 features sau encoding
```

3.7 Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Xử lý NaN/Inf sau scaling
X_train_scaled = np.nan_to_num(X_train_scaled, nan=0.0, posinf=0.0,
neginf=0.0)
```

3.8 Target Transformation

```
# Log transformation cho SalePrice (giảm skewness)
y = np.log1p(df['SalePrice'])

# Khi dự đoán, convert ngược:
predicted_price = np.expm1(prediction_log)
```

---

4. HUẤN LUYỆN VÀ ĐÁNH GIÁ MÔ HÌNH

4.1 Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

# Kết quả:
# Training set: 1,166 samples
# Test set: 292 samples
```

4.2 Các mô hình được huấn luyện

Traditional Machine Learning:

Model	Hyperparameters
Linear Regression	Default
Ridge Regression	alpha=10.0
Lasso Regression	alpha=0.001, max_iter=10000
ElasticNet	alpha=0.001, l1_ratio=0.5
Random Forest	n_estimators=100, random_state=42
Gradient Boosting	n_estimators=100, random_state=42

Deep Learning (Bonus):

```
# PyTorch Neural Network Architecture
HousePriceNN(
    Input(241)
    → Linear(256) → BatchNorm → ReLU → Dropout(0.3)
    → Linear(128) → BatchNorm → ReLU → Dropout(0.2)
    → Linear(64)  → BatchNorm → ReLU → Dropout(0.1)
    → Linear(32)  → ReLU
    → Linear(1)   # Output
)

# Training config:
# - Optimizer: Adam (lr=0.001)
# - Loss: MSELoss
# - Epochs: 100 (with early stopping, patience=15)
# - Batch size: 32
```

4.3 Metrics đánh giá

Metric	Công thức	Ý nghĩa
MAE	Σ|y - ŷ| / n	Sai số trung bình tuyệt đối
MSE	Σ(y - ŷ)² / n	Sai số bình phương TB
RMSE	√MSE	Căn bậc hai MSE
R² Score	1 - SS_res/SS_tot	Hệ số xác định (0-1)
CV R²	Cross-val R² (5-fold)	Đánh giá độ ổn định

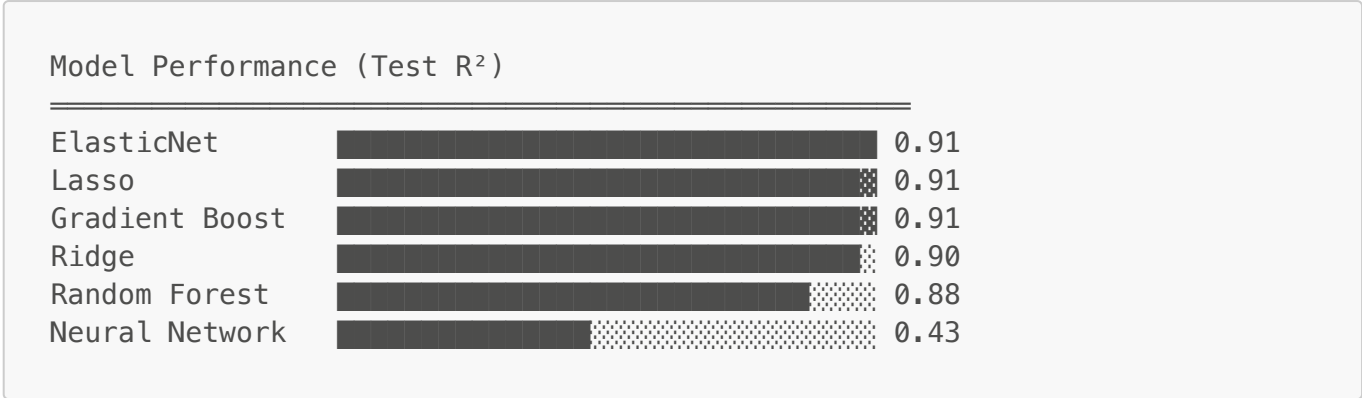
4.4 Kết quả so sánh mô hình

Bảng kết quả chi tiết:

Model	Train R²	Test R²	RMSE	MAE ($)	CV R²
ElasticNet	0.9471	**0.9097**	0.1234	$14,608	0.9031
Lasso	0.9447	0.9067	0.1254	$14,387	0.9075
Gradient Boosting	0.9653	0.9058	0.1260	$14,457	0.9067
Ridge	0.9484	0.9047	0.1268	$15,180	0.8961
Random Forest	0.9841	0.8821	0.1410	$15,944	0.8825
Linear Regression	0.9496	-6.2e14*	1.0e7*	$8.3M*	Unstable
Neural Network	-	0.4255	0.3112	$42,846	-

*Linear Regression bị multicollinearity, không ổn định.

Biểu đồ so sánh:



4.5 Best Model Selection

```
🏆 BEST MODEL: ElasticNet

Lý do lựa chọn:
✅ Test R² cao nhất: 0.9097
✅ RMSE thấp nhất: 0.1234
✅ CV R² ổn định: 0.9031 (±0.0266)
✅ Regularization (L1+L2) chống overfitting
✅ MAE chấp nhận được: $14,608
```

4.6 Phân tích Neural Network

Tại sao Neural Network performkém hơn?

1. **Dataset quá nhỏ:** 1,166 samples không đủ cho deep learning
2. **Bài toán tuyến tính:** Quan hệ giữa features và price gần như tuyến tính
3. **Feature engineering tốt:** Linear models tận dụng tốt hơn
4. **Overfitting:** 70,000+ parameters cho 1,166 samples

---

5. HƯỚNG DẪN SỬ DỤNG

5.1 Cài đặt môi trường

```
# Clone repository
git clone <repository_url>
cd "Final project"

# Tạo virtual environment
python3 -m venv venv
source venv/bin/activate  # Linux/Mac
# hoặc: venv\Scripts\activate  # Windows

# Cài đặt dependencies
pip install -r requirements.txt
```

5.2 Chạy Notebooks

```
# EDA Notebook
jupyter notebook notebooks/01_EDA.ipynb

# Training Notebook
jupyter notebook notebooks/02_Training.ipynb
```

5.3 Khởi động API Server

```
cd api
uvicorn main:app --reload --port 8000

# API sẽ chạy tại: http://localhost:8000
# Documentation: http://localhost:8000/docs
```

API Endpoints:

Method	Endpoint	Mô tả
GET	`/health`	Kiểm tra trạng thái API
GET	`/model-info`	Thông tin mô hình
GET	`/features`	Danh sách features
POST	`/predict`	Dự đoán giá nhà

Ví dụ request:

```
curl -X POST "http://localhost:8000/predict" \
  -H "Content-Type: application/json" \
  -d '{
    "OverallQual": 7,
    "GrLivArea": 1500,
    "YearBuilt": 2005,
    "YearRemodAdd": 2005,
    "FullBath": 2,
    "TotRmsAbvGrd": 7,
    "TotalBsmtSF": 1000,
    "GarageCars": 2,
    "GarageArea": 500
  }'
```

Response:

```
{
  "predicted_price": 186500.00,
  "predicted_price_formatted": "$186,500",
  "confidence_interval": {
    "lower": 158525.00,
    "upper": 214475.00,
    "formatted": "$158,525 - $214,475"
  },
  "model_info": {
    "model_name": "ElasticNet",
    "test_r2": 0.9097
  }
}
```

5.4 Khởi động Web App

```
cd app
streamlit run streamlit_app.py

# Web App sẽ chạy tại: http://localhost:8501
```

Tính năng Web App:

- 📝 Form nhập thông tin nhà
- 🔮 Dự đoán giá nhà real-time
- 📊 Hiển thị gauge chart và confidence interval
- 📋 Tóm tắt thông tin đã nhập

5.5 Quick Start Script

```
# Chạy script tự động
./run.sh

# Chọn option:
# 1) Run EDA Notebook
# 2) Run Training Notebook
# 3) Start API Server
# 4) Start Streamlit App
# 5) Start Both API and Streamlit
# 6) Exit
```

---

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Những gì đã đạt được:

✅ **Data Pipeline hoàn chỉnh:**

- Data validation với schema checking
- Feature engineering (14 features mới)
- Missing value handling
- Outlier detection và removal

✅ **Model Training:**

- So sánh 6 models traditional ML
- Thử nghiệm Deep Learning với PyTorch
- ElasticNet đạt $R^2 = 0.9097$

✅ **Deployment:**

- FastAPI backend RESTful API
- Streamlit frontend interactive
- Error handling và logging

✅ **Code Quality:**

- Modular architecture (src/, api/, app/)
- Type hints và documentation
- Consistent coding style

Key Insights:

1. **Feature Engineering quan trọng:** TotalSF, QualityArea là features mạnh nhất
2. **Linear models phù hợp:** Với dataset nhỏ và bài toán tuyến tính
3. **Regularization cần thiết:** ElasticNet outperform Linear Regression
4. **Deep Learning không phù hợp:** Dataset quá nhỏ cho neural networks

6.2 Hạn chế

Hạn chế	Mô tả
Dataset nhỏ	Chỉ 1,460 samples, hạn chế deep learning
Địa điểm cụ thể	Chỉ áp dụng cho Ames, Iowa
Temporal limitation	Data từ 2006-2010, có thể outdated
Missing features	Không có info trường học, crime rate

6.3 Hướng phát triển tương lai

Ngắn hạn:

- [ ] Hyperparameter tuning với GridSearchCV/Optuna
- [ ] Ensemble methods (Stacking, Voting)
- [ ] Feature selection (RFE, LASSO path)
- [ ] Cross-validation strategies (Stratified K-Fold)

Trung hạn:

- [ ] Thêm external data (school ratings, crime statistics)
- [ ] Time series features (housing market trends)
- [ ] Geospatial features (distance to amenities)
- [ ] A/B testing framework

Dài hạn:

- [ ] Triển khai production với Docker/Kubernetes
- [ ] Monitoring và model retraining pipeline
- [ ] Expand sang các thành phố khác
- [ ] Mobile application

6.4 Bài học kinh nghiệm

> **"Simple models often outperform complex ones on small datasets."**

1. **Understand your data first:** EDA là bước quan trọng nhất
2. **Feature engineering > Model complexity:** Đầu tư vào features
3. **Regularization prevents overfitting:** Luôn sử dụng regularization
4. **Validate with cross-validation:** Đừng chỉ dựa vào train/test split
5. **Deep Learning needs big data:** Không phải lúc nào cũng là solution

---

📚 TÀI LIỆU THAM KHẢO

1. Kaggle House Prices Competition: https://www.kaggle.com/c/house-prices-advanced-regression-techniques
2. Scikit-learn Documentation: https://scikit-learn.org/stable/
3. FastAPI Documentation: https://fastapi.tiangolo.com/
4. Streamlit Documentation: https://docs.streamlit.io/
5. PyTorch Documentation: https://pytorch.org/docs/

📎 PHỤ LỤC

A. Danh sách dependencies

```
pandas>=2.3.0
numpy>=2.4.0
scikit-learn>=1.8.0
torch>=2.9.0
fastapi>=0.128.0
streamlit>=1.52.0
```

B. Hardware requirements

- **Minimum:** 4GB RAM, 2-core CPU
- **Recommended:** 8GB RAM, 4-core CPU, GPU (optional for PyTorch)

C. Thời gian thực thi

Task	Thời gian
Data loading	~1s
Preprocessing	~2s
Model training (all models)	~30s
Neural Network (PyTorch)	~2-5 min
API response	<100ms

Báo cáo này được tạo tự động từ kết quả phân tích và huấn luyện mô hình.