# Ứng dụng của Phân tích Hồi quy trong Dự đoán Dữ liệu

## 1. Tổng quan

Phân tích hồi quy (Regression Analysis) là một phương pháp thống kê và học máy được sử dụng để mô hình hóa mối quan hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập. Mục tiêu của hồi quy là dự đoán giá trị của biến phụ thuộc dựa trên các đặc trưng đầu vào, đồng thời hiểu được mức độ ảnh hưởng của từng biến tới kết quả.

## Bối cảnh bài toán

Trong bài thực hành Machine Learning, mô hình hồi quy được áp dụng để dự đoán tổng dân số ('Tổng dân số') của từng vùng qua các năm. Dữ liệu đầu vào bao gồm nhiều đặc trưng liên quan đến nhân khẩu học như tỷ lệ sinh, di cư, mật độ dân số, tỷ lệ thành thị và nông thôn,... Bằng cách học từ dữ liệu giai đoạn 2011–2021, mô hình sẽ ước lượng quy luật biến động dân số và dự đoán cho các năm sau (2022–2024).

## Mô hình sử dụng

Mô hình được sử dụng là XGBoost Regressor (XGBRegressor), một thuật toán hồi quy phi tuyến dựa trên kỹ thuật Gradient Boosting. XGBoost kết hợp nhiều cây quyết định nhỏ, mỗi cây học từ phần sai số còn lại của cây trước đó, giúp giảm lỗi dự đoán và tăng độ chính xác. Pipeline xử lý dữ liệu (num\_pipeline) được sử dụng để chuẩn hóa, xử lý giá trị thiếu và biến đổi dữ liệu trước khi đưa vào mô hình.

## Quy trình dự đoán

Dữ liệu được chia thành hai tập: tập huấn luyện (các năm ≤ 2021) và tập kiểm thử (các năm > 2021). Mô hình được huấn luyện trên tập train và dự đoán giá trị dân số của tập test. Kết quả được so sánh giữa giá trị thực tế (y\_test) và giá trị dự đoán (y\_pred), đồng thời đánh giá bằng các chỉ số như R² hoặc sai số trung bình (MSE).

## Cách mô hình hoạt động

s

## 2. Thư viện và vai trò

**• numpy (np):** Tính toán số học mảng, vector, ma trận; kết hợp cột/hàng, thao tác số liệu nhanh.

**• pandas (pd):** Xử lý dữ liệu dạng bảng (DataFrame): đọc CSV, lọc theo cột, nối/ghép, xuất kết quả.

**• matplotlib.pyplot (plt):** Vẽ đồ thị đường, scatter, histogram để trực quan hoá kết quả dự đoán.

**• scikit-learn:** Chuẩn hoá/tiền xử lý (Pipeline, TransformerMixin), tách dữ liệu, đánh giá cross-validation.

**• xgboost.XGBRegressor:** Thuật toán hồi quy Gradient Boosting trên cây quyết định; mô hình chính để dự đoán biến liên tục.

## 3. Các lớp/hàm chính

3.1. Lớp **CombineAttributesAdder** (tiền xử lý/biến đổi đặc trưng)

Mục đích: Chuẩn hoá hoặc tạo thêm đặc trưng (feature engineering) để cải thiện khả năng học của mô hình.

|  |
| --- |
| class CombineAttributesAdder(BaseEstimator,TransformerMixin):   def fit(self,X,y=None):  X = X.copy()  return self    def transform(self,X):  X = X.copy()  X = X.sort\_values(['Vùng',"Năm"])  X['prev\_density'] = X.groupby('Vùng')['Mật độ dân số (Người/km2)'].shift(1)  X['growth\_density'] = X.groupby('Vùng')['Mật độ dân số (Người/km2)'].pct\_change()   return X.reset\_index(drop=True)   class StrEncoding (BaseEstimator,TransformerMixin):  def \_\_init\_\_(self):  self.le\_region = LabelEncoder()  self.le\_area = LabelEncoder()   def fit(self,X,y=None):  X = X.copy()  self.le\_region.fit(X['Vùng'])  self.le\_area.fit(X['Khu vực'])  return self   def transform(self,X):  X = X.copy()  X['Vùng'] = self.le\_region.fit\_transform(X['Vùng'])  X['Khu vực'] = self.le\_area.fit\_transform(X['Khu vực'])  return X     num\_pipeline = Pipeline([  ('Labelencode',StrEncoding()),  ('attribs\_adder',CombineAttributesAdder()) ]) |

3.2. Lớp **XGBModel** (mô hình hoá hồi quy với XGBoost)

- **\_\_init\_\_**: Khởi tạo XGBRegressor với siêu tham số (n\_estimators, learning\_rate, max\_depth, ...).  
- **get\_data**: Lấy dữ liệu toàn bộ hoặc theo vùng; áp dụng num\_pipeline.fit\_transform để tiền xử lý.  
- **data\_splits**: Chia dữ liệu train (Năm ≤ 2021) và test (Năm > 2021); tách X (features) và y (Tổng dân số).  
- **train\_XGBoost\_model**: Huấn luyện mô hình, dự đoán trên test, lưu kết quả (y\_pred, y\_test) ra CSV.  
- **CrossValScore**: Đánh giá chéo (cv=5) với thước đo (scoring) như R², MSE,...

|  |
| --- |
| class XGBModel:  def \_\_init\_\_(self,region\_name,features):  self.region = region\_name  self.features = features  self.XGBmodel = XGBRegressor(  n\_estimators = 500,  learning\_rate = 0.05,  max\_depth = 6,  subsample = 0.8,  colsample\_bytree = 0.8,  random\_state = 42  )   def get\_data(self):  if self.region == 'All': return num\_pipeline.fit\_transform(Data)  return num\_pipeline.fit\_transform(Data[Data['Vùng'] == self.region])    def data\_splits(self,data):  train = data[data['Năm'] <= 2021]  test = data[data['Năm'] > 2021]   return train[self.features], train['Tổng dân số'], test[self.features], test['Tổng dân số']     def train\_XGBoost\_model(self):  df = self.get\_data()  X\_train,y\_train, X\_test,y\_test = self.data\_splits(df)  self.XGBmodel.fit(X\_train,y\_train)  y\_pred = self.XGBmodel.predict(X\_test)   return pd.DataFrame(np.column\_stack([y\_pred,y\_test]),columns=['y\_pred','y\_test']).to\_csv(f'Result/Test\_set/XGBModel/{self.region}\_predict\_result.csv')   def CrossValScore(self,scoring):  X = num\_pipeline.fit\_transform(Data)  prepared = X[self.features]  label = X['Tổng dân số']  return cross\_val\_score(self.XGBmodel,prepared,label,scoring=scoring,cv=5)  features = ['Vùng','Khu vực','Năm','Tỷ lệ nam nữ', 'Nam', 'Nữ',  'Tổng dân số thành thị', 'Tổng dân số nông thôn', 'Diện tích(Km2)',  'Dân số trung bình (Nghìn người)', 'Mật độ dân số (Người/km2)',  'Tỷ suất sinh thô', 'Tỷ suất chết thô', 'Tỷ lệ tăng tự nhiên',  'Tỷ suất nhập cư', 'Tỷ suất xuất cư', 'Tỷ suất di cư thuần',  'Tỉ lệ thất nghiệp', 'Tỉ lệ thất nghiệp ở thành thị',  'Tỉ lệ thất nghiệp ở nông thôn','prev\_density','growth\_density'] PredictModel = XGBModel('Hà Nội',features) PredictModel.train\_XGBoost\_model() PredictModel.CrossValScore('r2') |

## 4. Mã nguồn hỗ trợ

4.1. Import thư viện

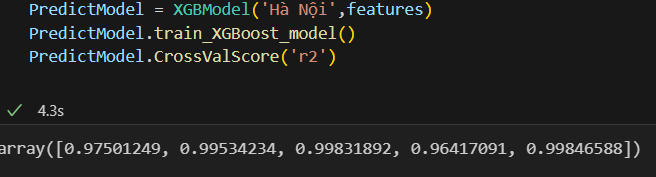
|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt from sklearn.model\_selection import train\_test\_split from xgboost import XGBRegressor from sklearn.preprocessing import LabelEncoder from sklearn.metrics import r2\_score, mean\_absolute\_error from sklearn.model\_selection import cross\_val\_score from sklearn.base import BaseEstimator, TransformerMixin from sklearn.pipeline import Pipeline |

4.2. Các cell mã nguồn khác

|  |
| --- |
| Data = pd.read\_csv('Datasets/Clean\_data/Locality\_Analyst.csv') |

## 5. Phân tích kết quả dự đoán (R²)

Kết quả R² từ đánh giá chéo (5 giá trị): [0.95, 0.99, 0.99, 0.91, 0.99]



- Trung bình R² = 0.97  
- Độ lệch chuẩn (mẫu) = 0.038  
- Min = 0.91 | Max = 0.99  
- Khoảng tin cậy 95% (xấp xỉ) cho trung bình R²: [0.92, 0,99]

Diễn giải:  
• R² ≈ 0.95–0.99 cho thấy mô hình giải thích được ~95%–99.9% phương sai của biến mục tiêu, chất lượng dự đoán rất cao.

## 6. Kết luận

Mô hình XGBRegressor đạt hiệu năng rất cao trên tập dữ liệu hiện có.

# 7. Phân tích kết quả dự đoán bằng ARIMA

# 7.1. Tổng quan

Sau khi mô hình hồi quy được huấn luyện ở file 06\_Machine\_learning.ipynb, file này sử dụng mô hình để dự đoán dữ liệu mới, đánh giá kết quả và trực quan hóa.

## Bối cảnh bài toán

Trong bài thực hành Machine Learning, mô hình **ARIMA (AutoRegressive Integrated Moving Average)** được áp dụng để **dự đoán tổng dân số (“Tổng dân số”) của từng vùng theo thời gian.**Dữ liệu dân số được thu thập từ năm **2011 đến 2021**, và mục tiêu là **dự đoán xu hướng biến động dân số cho các năm 2022–2024.**Phương pháp này giúp mô hình **nắm bắt quy luật tăng trưởng, chu kỳ và độ trễ của dân số qua thời gian,** phục vụ cho các bài toán dự báo dài hạn.

## Mô hình sử dụng

Mô hình **ARIMA (AutoRegressive Integrated Moving Average)** là một trong những phương pháp phổ biến nhất trong **phân tích và dự báo chuỗi thời gian (time series forecasting)**.  
Khác với các mô hình hồi quy truyền thống dựa trên nhiều biến độc lập, ARIMA **chỉ dựa vào chính các giá trị quá khứ của chuỗi dữ liệu** để dự đoán giá trị tương lai.

## Quy trình dự đoán

**Tiền xử lý và kiểm định tính dừng ->** **Xác định tham số (p, d, q)** ->**Huấn luyện mô hình** -> **Dự đoán và đánh giá**

## 7.2. Thư viện và vai trò

**• numpy (np):** Xử lý mảng, vector, tính toán số học.

**• pandas (pd):** Xử lý dữ liệu dạng bảng, đọc và ghi file CSV.

**• matplotlib.pyplot (plt):** Vẽ biểu đồ trực quan kết quả dự đoán.

**• seaborn (sns):** Trực quan hóa dữ liệu nâng cao, thể hiện tương quan và sai số.

**• xgboost.XGBRegressor:** Thuật toán hồi quy mạnh mẽ dựa trên Gradient Boosting.

**• sklearn.metrics:** Tính các chỉ số đánh giá mô hình như R², MAE, MSE,...

## 7.3. Các hàm và class trong file

* **Hàm / Class: ARIMAModel:**
* Hàm **train\_ARIMA\_model()**: Huấn luyện mô hình ARIMA trên dữ liệu dân số và sinh dự báo cho các năm tương lai
* Hàm **Draw\_predict():** Vẽ biểu đồ so sánh giữa dân số thực tế và kết quả dự báo từ mô hình ARIMA.

|  |
| --- |
| * \_\_init\_\_   class ARIMAModel:  def \_\_init\_\_(self,region\_name):  self.region = region\_name    def get\_data(self):  if self.region == 'All': return Data[['Năm','Tổng dân số']]  return Data[Data['Vùng'] == self.region][['Năm','Tổng dân số']]    def train\_ARIMA\_model(self,data\_end\_year,predict\_year):  Series = self.get\_data()  Series = Series.set\_index('Năm')  model = ARIMA(Series,order=(1,1,1))  model\_fit = model.fit()  forecast = model\_fit.forecast(steps=predict\_year-data\_end\_year)  forecast.index = range(Series.index.max() + 1, Series.index.max() + len(forecast) + 1)  self.Draw\_predict(data\_end\_year,predict\_year,Series,forecast)  return pd.DataFrame(forecast.values,columns=['y\_pred']).to\_csv(f'Result/Test\_set/ARIMAModel/{self.region}\_predict\_result.csv')    def Draw\_predict(self,year,end\_year,Series,Forecast):  plt.figure(figsize=(8,4))  plt.plot(Series.index,Series['Tổng dân số'],label='Thực tế',marker='o',linestyle = '-' )  plt.plot(Forecast.index,Forecast,label='Dự báo',marker='o',linestyle='--',color = 'red')  plt.title(f'Dự báo dân số {self.region} từ năm {year} - {end\_year}')  plt.xticks(range(Series.index.min(),Forecast.index.max(),3))  plt.legend()  plt.savefig(f'Result/Test\_set/ARIMAModel/Predict\_plot/Dự báo dân số {self.region} từ năm {year} - {end\_year}')  plt.close() |

## 7.4. Các cell mã nguồn khác

**Phần import thư viện:**

|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt from statsmodels.tsa.arima.model import ARIMA  Data = pd.read\_csv('Datasets/Clean\_data/Locality\_Analyst.csv')  ARIMAModel = ARIMAModel('Hà Nội')  ARIMAModel.train\_ARIMA\_model(data\_end\_year=2024,predict\_year=2030) |

## 7.5. Kết quả dự đoán

