HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
TP. HỒ CHÍ MINH

BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO
NĂM HỌC 2024-2025

DỰ BÁO GIÁ BẢO HIỂM BẰNG XGBOOST REGESSOR
MÃ SỐ ĐỀ TÀI : 07

Thuộc nhóm ngành khoa học: Điện tử

Tp. Hồ Chí Minh, ngày 27 tháng 10 năm 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
TP. HỒ CHÍ MINH

BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO
NĂM HỌC 2024-2025

DỰ BÁO GIÁ BẢO HIỂM BẰNG XGBOOST REGESSOR
MÃ SỐ ĐỀ TÀI : 07

Thuộc nhóm ngành khoa học: Điện tử

Sinh viên thực hiện:
Lâm Công Truyền N21DCDT096
Nguyễn Minh Chiến N21DCDT010
Đoàn Quang Hải N21DCDT027
Nguyễn Đức Tường N21DCDT099
Võ Phúc Nguyên N21DCDT063
Nguyễn Việt Hưng N21DCDT039

Người hướng dẫn: thầy giáo Hồ Nhựt Minh

Tp. Hồ Chí Minh, ngày 27 tháng 10 năm 2024

MỤC LỤC

TÓM TẮT

Dự án **"Dự báo giá bảo hiểm sử dụng XGBoost Regressor"** nhằm mục đích xây dựng một mô hình học máy dự đoán chi phí y tế của các công ty bảo hiểm dựa trên các đặc trưng như tuổi, chỉ số BMI, tình trạng hút thuốc, v.v. Điều này giúp các công ty bảo hiểm tính toán phí bảo hiểm chính xác hơn để đảm bảo lợi nhuận. Phương pháp truyền thống tính toán phí bảo hiểm thường tiêu tốn nhiều công sức và ngày càng trở nên phức tạp hơn với các mối quan hệ trong dữ liệu.
Trong dự án này, ngoài việc xây dựng mô hình **XGBoost Regressor**, còn sử dụng mô hình hồi quy tuyến tính để so sánh và học cách truyền tải kết quả kỹ thuật đến những người không chuyên.

1.GIỚI THIỆU

1.1 Tổng quan về dự án

Các công ty bảo hiểm chi trả các chi phí mà người được bảo hiểm phải chịu do thiệt hại về sức khỏe hoặc tài sản. Các chính sách thường được cung cấp bao gồm: hóa đơn y tế, nhà cửa, xe cơ giới, bảo hiểm cháy nổ, và các tổn thất tài chính như mất thu nhập. Người được bảo hiểm sẽ trả một khoản phí hoặc phí bảo hiểm cho công ty bảo hiểm để được hưởng các quyền lợi này. Các phương pháp truyền thống để tính toán phí bảo hiểm đòi hỏi rất nhiều lao động thủ công tốn thời gian và ngày càng trở nên phức tạp để có thể nắm bắt được các tương tác ngày càng phức tạp trong dữ liệu. Thông thường, các công ty bảo hiểm nên thu phí bảo hiểm cao hơn số tiền chi trả cho người được bảo hiểm nếu người đó yêu cầu bồi thường hợp lệ để tạo ra lợi nhuận. Vì lợi nhuận là yếu tố cơ bản giúp công ty bảo hiểm tồn tại, họ cần một cơ chế để dự báo chi phí chăm sóc sức khỏe một cách đáng tin cậy.
Do đó, mục tiêu của chúng ta là xây dựng một mô hình học máy giúp thiết lập mức phí bằng cách dự đoán các khoản phí hoặc chi trả do công ty bảo hiểm y tế thực hiện để duy trì lợi nhuận.
Trong dự án này, chúng ta chủ yếu tập trung vào việc xây dựng một mô hình XGBoost Regressor để xác định chi phí chăm sóc sức khỏe dựa trên các đặc trưng như tuổi tác, chỉ số khối cơ thể (BMI), hút thuốc, v.v. Chúng ta cũng sẽ tìm hiểu về tương quan phân loại, xây dựng mô hình hồi quy tuyến tính làm cơ sở và so sánh nó với kết quả của mô hình XGBoost Regressor. Cuối cùng, chúng ta sẽ học cách truyền đạt các kết quả kỹ thuật cho các bên liên quan không có chuyên môn kỹ thuật.

1.2 Mục tiêu và mục đích

Dự án này nhằm phát triển một API dự đoán chi phí bảo hiểm y tế dựa trên các thông tin cá nhân như tuổi, giới tính, chỉ số BMI, số con, tình trạng hút thuốc và vùng miền. API này sử dụng các mô hình học máy (machine learning) đã được huấn luyện trước để đưa ra dự đoán cho các yêu cầu mới từ người dùng.

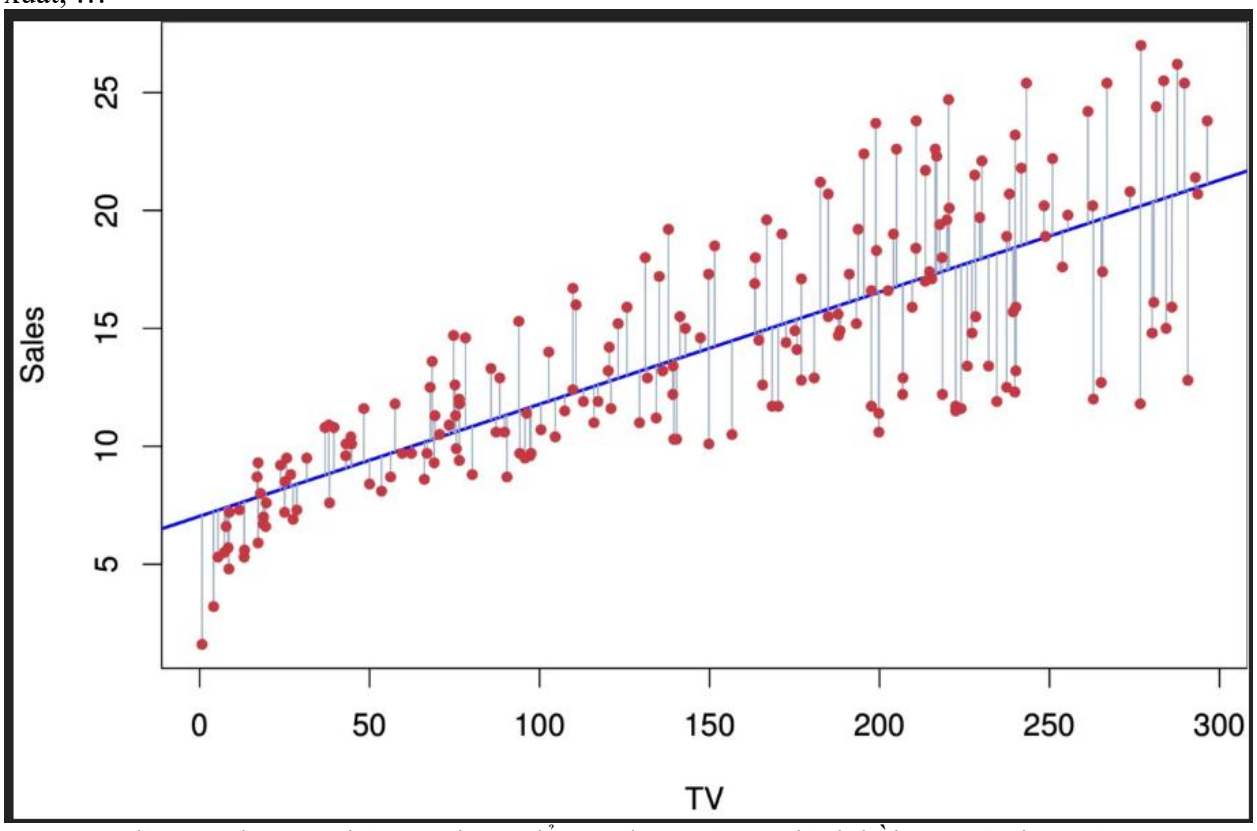
Mục tiêu chính của dự án là tạo ra một API mạnh mẽ có khả năng dự đoán chính xác chi phí bảo hiểm y tế dựa trên các thông tin cá nhân. Người dùng có thể tương tác với API này bằng cách gửi dữ liệu và nhận lại kết quả dự đoán một cách nhanh chóng.

1.3 Bối cảnh và nền tảng

1.3.1 Các mô hình học máy

Linear Regression là một thuật toán học có giám sát (supervised learning) trong Machine Learning, nó là một phương pháp thống kê dùng để ước lượng mối quan hệ giữa các biến độc lập (input features) và biến phụ thuộc (output target). Linear Regression giả định rằng sự tương quan giữa các biến là tuyến tính, từ đó tìm ra hàm tuyến tính tốt nhất để biểu diễn mối quan hệ này. Thuật toán này dự báo giá trị của biến output từ các giá trị của các biến đầu vào.

Mục tiêu của Linear Regression là tìm ra hệ số góc và điểm giao với trục tung sao cho hàm dự đoán tuyến tính đạt được sai số nhỏ nhất. Một trong những cách phổ biến để ước lượng các hệ số là sử dụng phương pháp Ordinary Least Squares (OLS), trong đó chúng ta cần tối thiểu hóa tổng bình phương sai số (sum of squared error).
Linear Regression được ứng dụng rộng rãi trong nhiều lĩnh vực, như:
Dự báo giá cả: dự đoán giá nhà, giá cổ phiếu, giá nhiên liệu dựa trên các yếu tố như vị trí, kích thước, chất lượng, lượng cung cầu, …
Dự báo điểm số: dự đoán điểm số của học sinh dựa trên thời gian học, nỗ lực, kỹ năng, trình độ giáo viên, …
Dự báo sản phẩm: dự đoán đầu ra sản xuất dựa trên thời gian, công suất, nguyên liệu, lao động, …
Phân tích chuỗi thời gian: dự đoán xu hướng và chu kỳ của các chuỗi dữ liệu, như bất động sản, thời tiết, xu hướng sản
xuất, …



XGBoost là một phương pháp mạnh mẽ để xây dựng các mô hình hồi quy có giám sát. Tính hợp lệ của tuyên bố này có thể được suy ra bằng cách biết về hàm mục tiêu (XGBoost) và các học viên cơ sở của nó. Hàm mục tiêu chứa hàm mất mát và một

thuật ngữ chính quy hóa. Nó cho biết về sự khác biệt giữa các giá trị thực tế và các giá trị dự đoán, tức là kết quả của mô hình cách xa các giá trị thực tế như thế nào. Các hàm mất mát phổ biến nhất trong XGBoost cho các bài toán hồi quy là reg:linear và đối với phân loại nhị phân là reg:logistics. Học tập tổng hợp bao gồm đào tạo và kết hợp các mô hình riêng lẻ (được gọi là học viên cơ sở) để có được một dự đoán duy nhất và XGBoost là một trong những phương pháp học tập tổng hợp. XGBoost mong đợi có các học viên cơ sở đều kém ở phần còn lại để khi tất cả các dự đoán được kết hợp, các dự đoán kém sẽ hủy bỏ và một dự đoán tốt hơn sẽ tổng hợp lại để tạo thành các dự đoán tốt cuối cùng.

1.3.2 Mô tả dữ liệu

Tập dữ liệu chứa thông tin về chi phí bảo hiểm y tế của các cá nhân ở Hoa Kỳ. Nó bao gồm các tính năng như tuổi, giới tính, chỉ số khối cơ thể (BMI), số lượng trẻ em, tình trạng hút thuốc và khu vực cư trú. Biến mục tiêu là "chi phí", đại diện cho chi phí y tế cá nhân được lập hóa đơn bởi bảo hiểm y tế.
Tập dữ liệu được lưu trữ trong tệp CSV có tên "insurance.csv". Mỗi hàng đại diện cho một cá nhân và mỗi cột đại diện cho một tính năng hoặc biến mục tiêu. Có tổng cộng 1338 hàng và 7 cột trong tập dữ liệu.

- age: Tuổi của người được hưởng bảo hiểm chính.
- sex: Giới tính của người được hưởng bảo hiểm chính.
- bmi: Chỉ số khối cơ thể của người được hưởng bảo hiểm chính.
- children: Số con của người được hưởng bảo hiểm chính.
- smoker: Người được hưởng bảo hiểm chính có hút thuốc hay không.
- region: Khu vực cư trú của người được hưởng bảo hiểm chính tại Hoa Kỳ.
- charges: Chi phí y tế cá nhân được thanh toán bởi bảo hiểm y tế (đây là biến mục tiêu, biến cần dự đoán).

2. YÊU CẦU PHẦN MỀM

- Lập trình: Colab, Python, Visual studio code, Jupyter notebook
- Flask**:** Framework để xây dựng và triển khai API.
- Thư viện:
  - pandas: dùng để đọc, xử lý và phân tích dữ liệu dạng bảng.
  - numpy: dùng cho các phép toán số học và xử lý mảng.
  - matplotlib và seaborn: dùng để trực quan hóa dữ liệu.
  - scikit-learn (sklearn): dùng cho các tác vụ học máy, bao gồm:
  - train_test_split: chia dữ liệu thành tập huấn luyện và tập kiểm tra.
  - LinearRegression: xây dựng mô hình hồi quy tuyến tính.
  - metrics: đánh giá hiệu suất mô hình (ví dụ: RMSE).
  - Pipeline: xây dựng pipeline cho quy trình huấn luyện.
  - xgboost: dùng để xây dựng mô hình XGBoost Regressor.
  - statsmodels: dùng cho thống kê và kiểm định giả thuyết (ví dụ: ANOVA).
  - scipy: dùng cho tính toán khoa học (ví dụ: kiểm định Chi-bình phương).

3. KIẾN TRÚC

3.1 Lựa chọn mô hình

3.1.1 XGBoost (eXtreme Gradiant Boost)

XGBoost (eXtreme Gradiant Boost) là một thuật toán dựa trên cây quyết định, tăng cường độ dốc. Đây là thuật toán hoạt động hàng đầu trong nhiều cuộc thi Kaggle, vì vậy là một ứng cử viên lý tưởng để áp dụng cho trường hợp sử dụng này.

XGBoost cho hồi quy được đào tạo bằng cách sử dụng quy trình sau:

1. Đưa ra dự đoán ban đầu (thường chỉ là giá trị 0.5)
2. Tính dư của mỗi quan sát (so với dự đoán ban đầu này)
3. Lắp cây quyết định hồi quy vào phần dư:

3.1. Tính toán điểm tương đồng cho các quan sát trong mỗi lá:

- Công thức: $S = \sum \frac{(R)^2}{N_R + \lambda}$
- Trong đó:
  S: Điểm tương đồng.
  $\Sigma(R)^2$: Tổng bình phương của các giá trị phần dư (residual).
  NR**:** Số lượng phần dư.
  λ: Tham số điều chuẩn (regularization parameter).

3.2. Thử phân chia các nút lá hơn nữa để cải thiện việc phân cụm các phần dư tương tự:

- Đối với mỗi lần phân chia, điểm tương đồng được tính cho các nút lá trái và phải.
- Sự cải thiện bằng cách phân chia được định lượng bằng cách sử dụng độ lợi (gain): $G = S_{left} + S_{right} - S_{root}$
- Chọn phân chia tối đa hóa độ lợi.

3.3.Lặp lại ba bước này cho đến khi đạt đến số lượng quan sát tối thiểu trên mỗi lá (xem tham số trong XGBoost API).

4. Cây quyết định hồi quy sau đó được cắt tỉa dựa trên giá trị độ lợi (gain) của mỗi lần phân chia, sử dụng ngưỡng γ (tham số gamma trong XGBoost API):

4.1.Đối với lần phân chia thấp nhất trong cây, nếu G < γ, thì lần phân chia đó sẽ bị loại bỏ; nếu nó dương, lần phân chia đó sẽ được giữ lại.

4.2.Lặp lại quá trình này cho mỗi lần phân chia, di chuyển lên phía gốc (và bao gồm cả chính gốc cây).

5. Sau khi cây quyết định hồi quy được cắt tỉa, hãy tính toán giá trị đầu ra của mỗi lá: $O = \sum \frac{(R)}{N_R + \lambda}$

5.1.Lưu ý: Khi λ > 0, nó sẽ làm giảm lượng mà một lá riêng lẻ thêm vào dự đoán tổng thể, làm giảm độ nhạy của dự đoán cuối cùng đối với dự đoán của cây riêng lẻ (xem tham số reg_lambda trong XGBoost API).

6. Sau đó, kết hợp dự đoán ban đầu với dự đoán từ cây quyết định hồi quy (được điều chỉnh theo hệ số học, η - xem tham số learning_rate trong XGBoost API) để có được dự đoán mới: Prednew = Predinit + η * Predresidual.

7. Lặp lại các bước từ 1-6 cho đến khi phần dư đạt đến một ngưỡng nhất định hoặc cho đến khi đạt đến số lượng cây tối đa (xem tham số n_estimators trong XGBoost API).

3.1.2 Linear Regression

Hồi quy tuyến tính (Linear Regression): Là một phương pháp thống kê được sử dụng để mô hình hóa mối quan hệ giữa hai biến số. Mối quan hệ này được giả định là tuyến tính, nghĩa là sự thay đổi của một biến số sẽ dẫn đến sự thay đổi tương ứng của biến số kia theo một tỷ lệ cố định. Hồi quy tuyến tính được biểu diễn trực quan dưới dạng một đường thẳng, với độ dốc xác định cách thay đổi của một biến ảnh hưởng đến thay đổi của biến kia. Giao điểm y của mối quan hệ hồi quy tuyến tính phản ánh giá trị của một biến khi giá trị của biến kia bằng 0.
Hồi quy tuyến tính bội ước lượng mối quan hệ giữa hai hoặc nhiều biến độc lập và một biến phụ thuộc.
$\hat{y} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \varepsilon$
Trong đó:

- p là số lượng đặc trưng (features) trong mô hình.
- Đối với bất kỳ giá trị biến độc lập (x) nào cho trước, y là giá trị dự đoán của biến phụ thuộc (y).
- β_0 đại diện cho giao điểm, hoặc giá trị dự đoán của y khi x bằng 0.
- β_1 là hệ số hồi quy của biến x_1, cho biết y sẽ thay đổi bao nhiêu khi x_1 tăng hoặc giảm.
- β_p là hệ số hồi quy của biến cuối cùng x_p, cho biết y sẽ thay đổi bao nhiêu khi x_p tăng hoặc giảm.
- x_1 ... x_p là các biến độc lập hoặc biến dự đoán giúp chúng ta dự đoán y.

4. THIẾT KẾ CHI TIẾT MÔ HÌNH HỌC MÁY

Mô hình tuyến tính cơ sở (Baseline Linear Model):
Thuật toán: Hồi quy tuyến tính (Linear Regression)
Mục đích:

- Dùng làm điểm chuẩn để so sánh với các mô hình phức tạp hơn.
- Hiểu rõ hơn về mối quan hệ tuyến tính giữa các biế

Các bước thực hiện:

- Kiểm tra các giả định của hồi quy tuyến tính (ví dụ: tính tuyến tính, độc lập, phương sai đồng nhất, phân phối chuẩn của phần dư).
- Tiền xử lý dữ liệu: Có thể bao gồm mã hóa biến phân loại, chuẩn hóa hoặc biến đổi các biến số.
- Huấn luyện mô hình bằng cách sử dụng hàm từ thư viện .LinearRegressionsklearn.linear_model
- Đánh giá mô hình dựa trên các metrics như RMSE, R-squared, v.v.

Mô hình XGBoost:

Thuật toán: XGBoost Regressor
Mục đích: Cải thiện độ chính xác dự đoán so với mô hình tuyến tính cơ sở.
Các bước thực hiện:

- Tiền xử lý dữ liệu: Tương tự như mô hình tuyến tính, nhưng có thể bao gồm các kỹ thuật phức tạp hơn như , .StandardScalerPowerTransformer
- Sử dụng của Sklearn để tối ưu hóa quá trình huấn luyện mô hình.Pipeline
- Tối ưu siêu tham số bằng cách sử dụng với .BayesSearchCVskopt
- Đánh giá mô hình dựa trên các metrics như RMSE, R-squared, và so sánh với mô hình tuyến tính cơ sở.

Thiết kế chi tiết của mô hình XGBoost:

Phần quan trọng nhất của dự án này là việc sử dụng XGBoost Regressor để cải thiện độ chính xác dự đoán. Thiết kế chi tiết của mô hình này bao gồm:

- Tiền xử lý: Dữ liệu được chuẩn hóa bằng và có thể được biến đổi bằng để cải thiện hiệu suất của mô hình.StandardScalerPowerTransformer
- Lựa chọn đặc trưng: Có thể sử dụng (Recursive Feature Elimination) để lựa chọn các đặc trưng quan trọng nhất.RFE
- Huấn luyện: Mô hình được huấn luyện bằng cách sử dụng hàm từ thư viện .XGBRegressorxgboost
- Tối ưu siêu tham số: được sử dụng để tìm kiếm các giá trị tối ưu cho các siêu tham số của XGBoost, chẳng hạn như , , , , , , v.v.BayesSearchCVlearning_raten_estimatorsmax_depthmin_child_weightgammareg_lambda
- Đánh giá: Hiệu suất của mô hình được đánh giá dựa trên RMSE và được so sánh với mô hình tuyến tính cơ sở.

4.1 The input

Dữ liệu: Tập dữ liệu chứa thông tin về các cá nhân, bao gồm:insurance.csv

- age: Tuổi của người được hưởng bảo hiểm chính.
- sex: Giới tính của người được hưởng bảo hiểm chính.
- bmi: Chỉ số khối cơ thể của người được hưởng bảo hiểm chính.
- children: Số con của người được hưởng bảo hiểm chính.
- smoker: Người được hưởng bảo hiểm chính có hút thuốc hay không.
- region: Khu vực cư trú của người được hưởng bảo hiểm chính tại Hoa Kỳ.
- charges: Chi phí y tế cá nhân được thanh toán bởi bảo hiểm y tế (đây là biến mục tiêu, biến cần dự đoán).

4.2 The output

- Mô hình được huấn luyện: Một mô hình XGBoost Regressor đã được huấn luyện trên dữ liệu đầu vào và được tối ưu hóa với các siêu tham số đã chọn. Mô hình này có khả năng dự đoán chi phí y tế () cho một cá nhân mới dựa trên các thông tin cá nhân của họ.charges
- Đánh giá hiệu suất: Các chỉ số đánh giá hiệu suất của mô hình, chủ yếu là RMSE (Root Mean Squared Error), được sử dụng để đo lường độ chính xác

của mô hình trong việc dự đoán chi phí y tế. Các chỉ số khác như R-squared cũng có thể được sử dụng.

- Thông tin chi tiết: Các thông tin chi tiết về mô hình, chẳng hạn như tầm quan trọng của các đặc trưng, có thể được trích xuất để hiểu rõ hơn về cách thức mô hình hoạt động và các yếu tố nào ảnh hưởng nhiều nhất đến chi phí y tế.

4.3 Tiền xử lý

Xử lý biến phân loại:

- Các biến phân loại , và được chuyển đổi thành dạng số bằng cách sử dụng One-Hot Encoding. sexsmokerregion
- One-Hot Encoding tạo ra các biến giả (dummy variables) mới, mỗi biến đại diện cho một giá trị duy nhất của biến phân loại.
- Pandas được sử dụng cho One-Hot Encoding.get_dummies

Chuẩn hóa:

- Các biến số , , và được chuẩn hóa bằng để chúng có trung bình bằng 0 và độ lệch chuẩn bằng 1.agebmichildrenStandardScaler
- Chuẩn hóa giúp cải thiện hiệu suất của mô hình XGBoost, đặc biệt khi các biến có thang đo khác nhau.

Biến đổi (Tùy chọn):

- Biến đổi được áp dụng cho biến (biến mục tiêu) để giảm độ lệch (skewness).PowerTransformercharges
- Biến đổi giúp dữ liệu tuân theo phân phối chuẩn hơn, có thể cải thiện hiệu suất của mô hình tuyến tính.
- Bước này được coi là tùy chọn và có thể được bỏ qua dựa trên phân tích dữ liệu và hiệu suất của mô hình.

5. TRIỂN KHAI DỰ ÁN

5.1 Thêm thư viện cần thiết:

Trước khi bắt đầu, hãy import 3 thư viện cần thiết:

```
# import libraries
import pandas as pd
import numpy as np
import plotly.express as px
import sys
from sklearn.model_selection import train_test_split
from category_encoders import OneHotEncoder
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import math
from xgboost import XGBRegressor
from sklearn.pipeline import Pipeline
```

```
from skopt import BayesSearchCV
from skopt.space import Real, Categorical, Integer
from sklearn.preprocessing import StandardScaler, PowerTransformer
from sklearn.feature_selection import RFE
import projectpro
projectpro.checkpoint('1e808c')
```

-**Pandas**: Được sử dụng để thao tác và phân tích dữ liệu.
-**Numpy**: Được sử dụng để thực hiện các phép tính số và làm việc với vùng mảng.
-**Plotly**: sử dụng các để tạo các biểu đồ và hình ảnh hóa tương tác.
- Sys: model này cung cấp quyền truy cập một số biến và hàm được sử dụng hoặc duy trì python.
- **sklearn.model_selection import train_test_split**: từ scikit-learn. Hàm này được sử dụng để chia dữ liệu thành các tập huấn luyện và thử nghiệm.
-**from category_encoders import OneHotEncoder**: Nhập lớp OneHotEncoder từ category_encoders. Bộ mã hóa này được sử dụng để chuyển đổi các biến phân loại thành một biểu diễn số.
-from sklearn.linear_model import LinearRegression: Nhập lớp LinearRegression từ scikit-learn. Lớp này được sử dụng để thực hiện hồi quy tuyến tính.
-import statsmodels.api as sm: Nhập mô-đun statsmodels.api và đặt bí danh là 'sm'. Statsmodels được sử dụng để ước lượng thống kê và kiểm định giả thuyết
-from sklearn.pipeline import Pipeline: Nhập lớp Pipeline từ scikit-learn. Lớp này được sử dụng để xâu chuỗi nhiều bước ước lượng lại với nhau.
-from skopt import BayesSearchCV: Nhập lớp BayesSearchCV từ scikit-optimize. Lớp này được sử dụng để thực hiện tối ưu hóa siêu tham số dựa trên Bayes.
from skopt.space import Real, Categorical, Integer:
Nhập các lớp Real, Categorical và Integer từ scikit-optimize. Các lớp này được sử dụng để xác định không gian tìm kiếm cho các siêu tham số.
-from sklearn.preprocessing import StandardScaler, PowerTransformer: Nhập các lớp StandardScaler và PowerTransformer từ scikit-learn. Các lớp này được sử dụng để mở rộng quy mô và biến đổi các đặc trưng.
-from sklearn.feature_selection import RFE: Nhập lớp RFE (Recursive Feature Elimination) từ scikit-learn. Lớp này được sử dụng để chọn đặc trưng.
-import projectpro: Nhập mô-đun **projectpro**. Mô-đun này có khả năng đến từ ProjectPro, một nền tảng giáo dục khoa học dữ liệu, và có thể được sử dụng cho các mục đích cụ thể của nền tảng đó.
-projectpro.checkpoint('1e808c'): Gọi hàm checkpoint từ mô-đun projectpro. Hàm này có khả năng được **ProjectPro** sử dụng để theo dõi tiến trình hoặc tạo điểm kiểm tra trong một dự án
Và **funtions** ở thư mục **ml_pipeline**:

```
# ml_pipeline folder should be in the same directory as the notebook, append the path and import functions
sys.path.append('ml_pipeline/')
from eda import plot_histograms, plot_univariate_numeric, plot_univariate_categorical, \
plot_heatmap, plot_paired_boxplots, plot_paired_scatterplots, plot_residuals, plot_pearson_wrt_target
from stats import chi2, anova
from model_performance import calc_model_performance, compare_model_performance, calc_preds_in_residual_range,\
```

```
calc_preds_in_residual_perc_range
```

-sys.path.append(...): Dòng này sửa đổi đường dẫn hệ thống Python để bao gồm thư mục chứa thư mục 'ml_pipeline'. Điều này đảm bảo rằng Python có thể tìm và nhập các mô-đun tùy chỉnh trong thư mục đó. Đường dẫn cụ thể này là đường dẫn trên hệ thống tệp của người dùng.

-from eda import ...: Nhập một số hàm từ mô-đun (có thể có tên 'eda.py') trong thư mục 'ml_pipeline'. Các hàm này có thể được sử dụng cho các tác vụ EDA:

-plot_histograms: Có thể tạo biểu đồ histogram để trực quan hóa dữ liệu.

-plot_univariate_numeric: Có thể tạo biểu đồ cho các đặc trưng số liên quan đến biến mục tiêu.

-plot_univariate_categorical: Có thể tạo biểu đồ cho các đặc trưng phân loại liên quan đến biến mục tiêu.

-plot_heatmap: Có thể tạo biểu đồ heatmap để trực quan hóa mối tương quan giữa các đặc trưng số.

-plot_paired_boxplots: Có thể tạo biểu đồ boxplot để so sánh các nhóm dữ liệu.

-plot_paired_scatterplots: Có thể tạo biểu đồ scatterplot để trực quan hóa mối quan hệ giữa các đặc trưng số.

-plot_residuals: Có thể tạo biểu đồ để phân tích phần dư của mô hình.

-plot_pearson_wrt_target: Có thể tạo biểu đồ thể hiện mối tương quan Pearson với biến mục tiêu.

-from stats import chi2, anova: Nhập các hàm có thể liên quan đến kiểm định thống kê:

-chi2: Có thể triển khai hoặc thực hiện kiểm định chi bình phương.

-anova: Có thể triển khai hoặc thực hiện kiểm định phân tích phương sai (ANOVA).

-from model_performance import ...: Nhập các hàm để đánh giá hiệu suất mô hình:

-calc_model_performance: Có thể tính toán các chỉ số hiệu suất khác nhau cho một mô hình.

-compare_model_performance: Có thể so sánh các chỉ số hiệu suất giữa hai mô hình.

-calc_preds_in_residual_range: Có thể tính toán số lượng dự đoán nằm trong một khoảng phần dư cụ thể.

-calc_preds_in_residual_perc_range: Có thể tính toán số lượng dự đoán nằm trong một khoảng phần trăm phần dư cụ thể.

5.2 Phân tích dữ liệu thăm dò (EDA)

- Đầu tiên, hãy đọc trong tập dữ liệu được lưu trữ trong tệp

```
data = pd.read_csv('../data/insurance.csv')
```

-Dòng này đọc một tệp CSV có tên 'insurance.csv' nằm trong thư mục 'data' và lưu trữ nó vào một pandas DataFrame có tên là data. Đây là cách bạn tải dữ liệu vào môi trường để phân tích

-Bây giờ chúng ta có thể ước lượng tập dữ liệu:

```
data.head()
```

-Dòng này hiển thị một vài hàng đầu tiên của DataFrame data, cho phép bạn nhanh chóng kiểm tra cấu trúc và nội dung của dữ liệu. Đây là cách tốt để có cái nhìn đầu tiên về dữ liệu của

bạn.

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Hãy trả về kiểu dữ liệu của các cột:

```
data.info()
```

-Dòng này cung cấp một bản tóm tắt về DataFrame, bao gồm kiểu dữ liệu của mỗi cột, số lượng giá trị không phải là null và mức sử dụng bộ nhớ. Nó hữu ích để hiểu các đặc điểm cơ bản của dữ liệu.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

-Vì vậy, chúng ta có ba đặc điểm số (`age`, `bmi` và `children`) và ba đặc điểm phân loại (`sex`, `smoker` và `region`).
-Mục tiêu (tức là biến mà chúng ta muốn dự đoán) là cột `charges`, vì vậy hãy chia tập dữ liệu thành các đặc điểm (`X`) và mục tiêu (`y`):

```
target = 'charges'
X = data.drop(target, axis=1)
y = data[target]
```

-target = 'charges': Dòng này gán chuỗi 'charges' cho biến target, cho biết rằng cột 'charges' trong DataFrame là biến mục tiêu (biến mà chúng ta muốn dự đoán)
-X = data.drop(target, axis=1): Dòng này tạo một DataFrame X mới bằng cách loại bỏ cột 'charges' (biến mục tiêu) khỏi DataFrame data ban đầu. X bây giờ chỉ chứa các đặc trưng (biến đầu vào) cho mô hình.
-y = data[target]: Dòng này tạo một pandas Series y chỉ chứa các giá trị từ cột 'charges'. Đây là dữ liệu biến mục tiêu của bạn, tách biệt với các đặc trưng.
-Kiểm tra hình dạng khung dữ liệu:

```
X.shape, y.shape
```

-X.shape, y.shape: Dòng này hiển thị kích thước (số hàng và số cột) của DataFrame/Series X (đặc trưng) và y (biến mục tiêu). Đây là cách để kiểm tra kích thước của các cấu trúc dữ liệu của bạn.

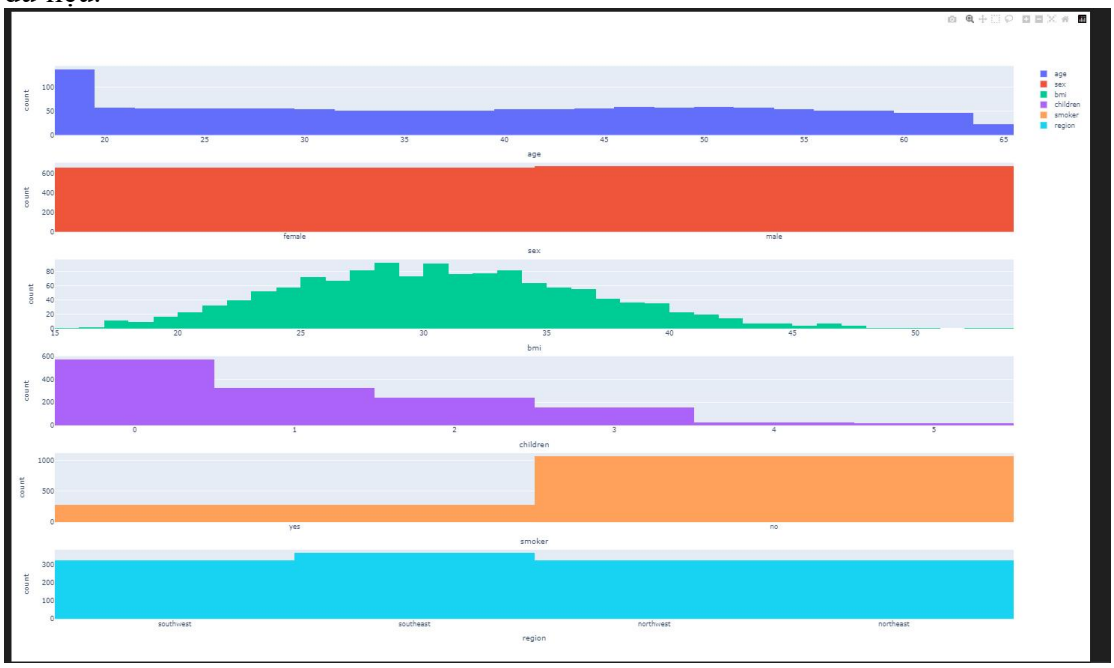
```
...   ((1338, 6), (1338,))
```

5.1.1. Phân phối

Bây giờ chúng ta hãy xem xét phân phối của từng đặc điểm bằng cách vẽ biểu đồ histogram cho từng đặc điểm:

```
# Plots histogram for each feature using plotly library
plot_histograms(X)
```

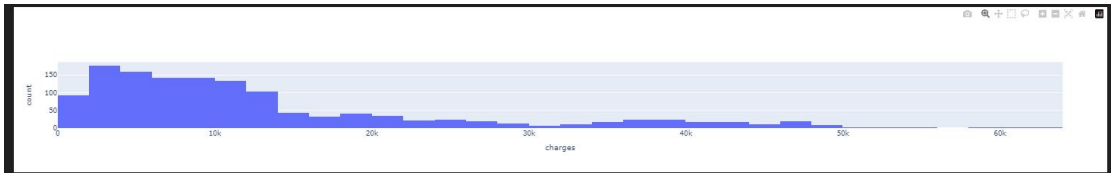
-Dòng này gọi một hàm tùy chỉnh plot_histograms (có thể được định nghĩa trong tệp 'ml_pipeline/eda.py') để tạo biểu đồ histogram cho từng đặc trưng (cột) trong DataFrame X. Biểu đồ histogram là một cách hữu ích để trực quan hóa phân phối của dữ liệu.



Những điểm cần lưu ý liên quan đến sự phân bố của từng đặc điểm:
Age - Phân bố gần như đồng đều.
Sex - Thể tích gần như bằng nhau trong mỗi danh mục.
bmi - Phân bố gần như bình thường.
Children- Lệch phải (tức là thể tích cao hơn trong phạm vi thấp hơn).
Smoker - Thể tích lớn hơn đáng kể trong danh mục `không` so với danh mục `có`.
Region - Thể tích gần như bằng nhau trong mỗi danh mục.

```
# Plots histogram for target using plotly library
plot_histograms(pd.DataFrame(y), height=300)
```

-Dòng này gọi cùng một hàm plot_histograms nhưng dành riêng cho biến mục tiêu y. Nó chuyển đổi y (một Series) thành DataFrame và đặt chiều cao tùy chỉnh cho biểu đồ histogram. Điều này giúp trực quan hóa phân phối của biến mục tiêu.



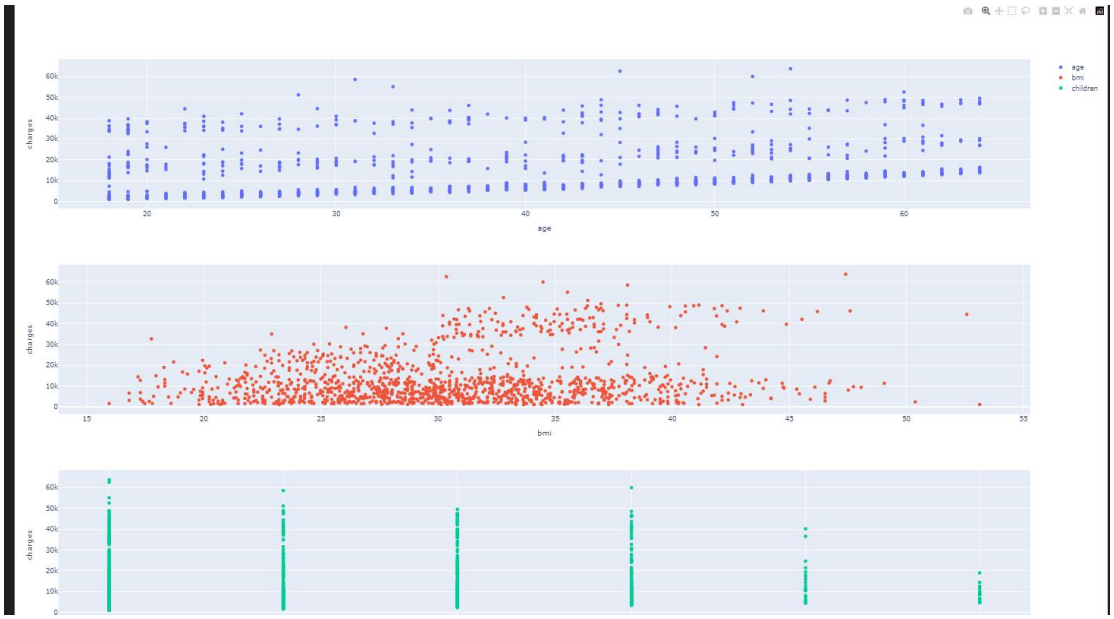
Các phân phối lệch về bên phải (tức là có khối lượng lớn hơn ở phía thấp).

5.2.2.Phân tích đơn biến (liên quan đến mục tiêu)

Các tính năng số

```
plot_univariate_numeric(
    X.select_dtypes(include=np.number),
    y
)
```

-X.select_dtypes(include=np.number): Phần này của mã chọn lọc chỉ các đặc trưng số (các cột có kiểu dữ liệu số) từ DataFrame X. Việc này được thực hiện bằng phương thức select_dtypes của pandas **DataFrames**, cho phép bạn lọc các cột dựa trên kiểu dữ liệu của chúng. Bằng cách chỉ định **include=np.number**, nó chọn các cột có kiểu dữ liệu như số nguyên (int64) và số thực (float64).
-y: Biểu diễn biến mục tiêu (biến mà bạn muốn dự đoán), có thể được lưu trữ trong pandas Series hoặc một cột của DataFrame. Đây là biến mà các đặc trưng số sẽ được phân tích dựa trên nó.
-plot_univariate_numeric(...): Đây là một hàm tùy chỉnh mà bạn có thể đã định nghĩa trong tệp **ml_pipeline/eda.py** (hoặc một vị trí tương tự). Nó nhận các đặc trưng số và biến mục tiêu làm đầu vào. Mục đích của hàm này có thể là tạo ra các hình ảnh hóa hiển thị mối quan hệ giữa mỗi đặc trưng số và biến mục tiêu.



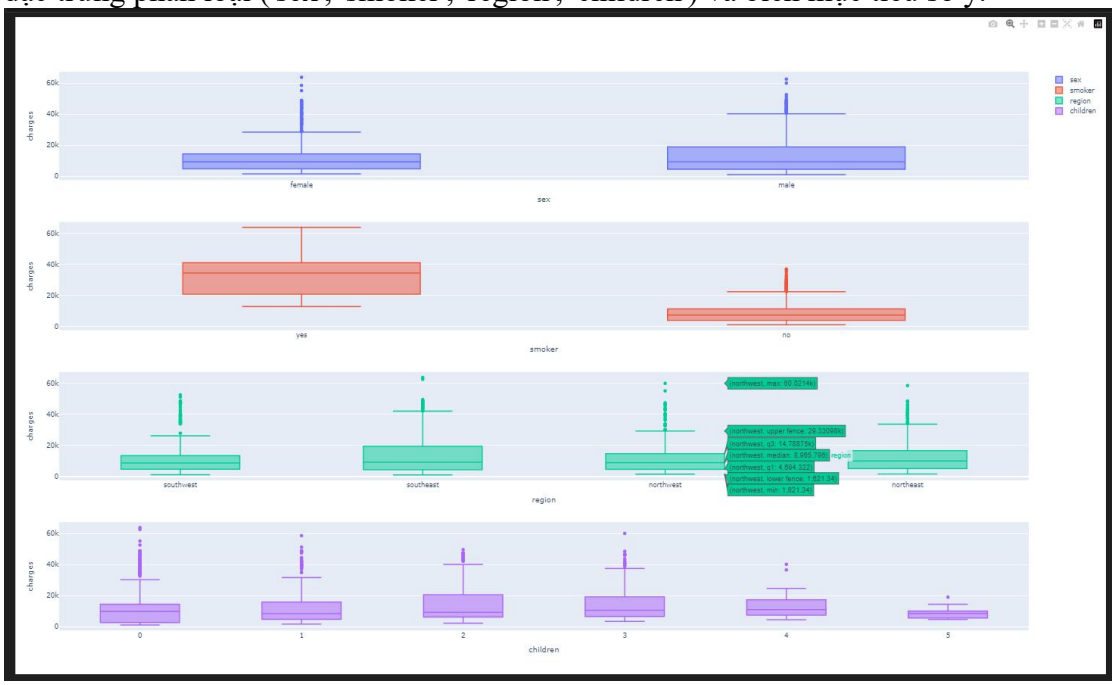
Những điểm cần lưu ý liên quan đến từng tính năng:
-age`** - Khi `age**` tăng, `**charges**` cũng có xu hướng tăng (mặc dù có sự khác biệt lớn về `**charges**` đối với một `**age**` nhất định).

bmi - Không có mối quan hệ rõ ràng, mặc dù có vẻ như có một nhóm cá nhân có `bmi` > 30 có `charges` > 30k. Nhóm này có thể trở nên rõ ràng hơn khi chúng ta thực hiện phân tích hai biến sau này.
-children` - Không có mối quan hệ rõ ràng (mặc dù `charges` có vẻ giảm khi `children` tăng). Vì chỉ có 6 giá trị duy nhất cho tính năng này, chúng ta hãy thử coi nó như một tính năng phân loại cho mục đích phân tích đơn biến.

Tính năng phân loại

```
plot_univariate_categorical(
    X[['sex', 'smoker', 'region', 'children']],
    y
)
```

-Đoạn mã này có mục đích tạo ra các biểu đồ để trực quan hóa mối quan hệ giữa các đặc trưng phân loại ('sex', 'smoker', 'region', 'children') và biến mục tiêu số y.



-Những điểm cần lưu ý liên quan đến từng tính năng:
+ `sex` - Không có sự khác biệt đáng kể nào về `charges` giữa các danh mục.
+ `smoker` - `charges` cho `smoker` == 'có' thường cao hơn nhiều so với khi `smoker` == 'không'.
+ `region` - Không có sự khác biệt đáng kể nào về `charges` giữa các danh mục.
+ `children` - Không có sự khác biệt đáng kể nào về `charges` giữa các danh mục (`children` >= 4 có xu hướng nghiêng về `charges` thấp hơn, nhưng điều này có thể là do khối lượng thấp trong các danh mục đó - hãy xem phần **Phân phối**).

5.2.3.Phân tích hai biến (liên quan đến mục tiêu)

-Bây giờ chúng ta hãy áp dụng phân tích hai biến liên quan đến mục tiêu. Điều này có nghĩa là chúng ta lấy các cặp tính năng và xem chúng liên quan như thế nào đến mục tiêu.

-Cách chúng ta thực hiện điều này phụ thuộc vào việc các cặp tính năng đều là số, đều là danh mục hay hỗn hợp giữa danh mục và số. Đối với tất cả các cặp số, chúng ta sẽ sử dụng bản đồ nhiệt; đối với tất cả các cặp danh mục, chúng ta sẽ sử dụng biểu đồ hộp; đối với các cặp số phân loại, chúng ta sẽ sử dụng biểu đồ phân tán.

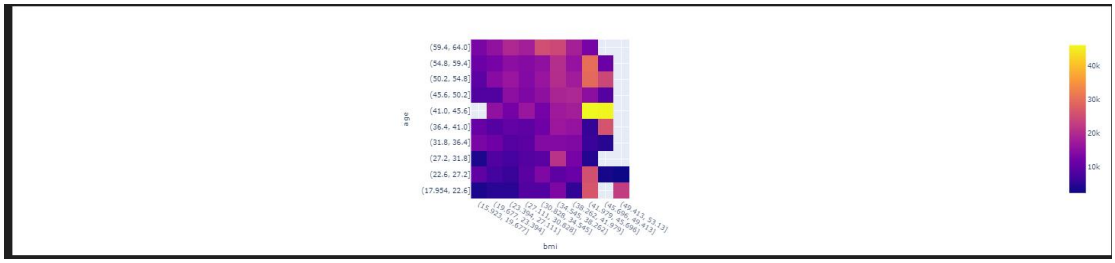
.Cặp số

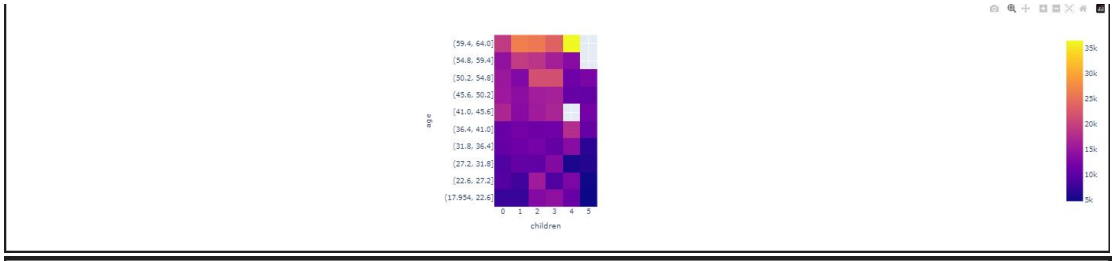
Bản đồ nhiệt tương quan

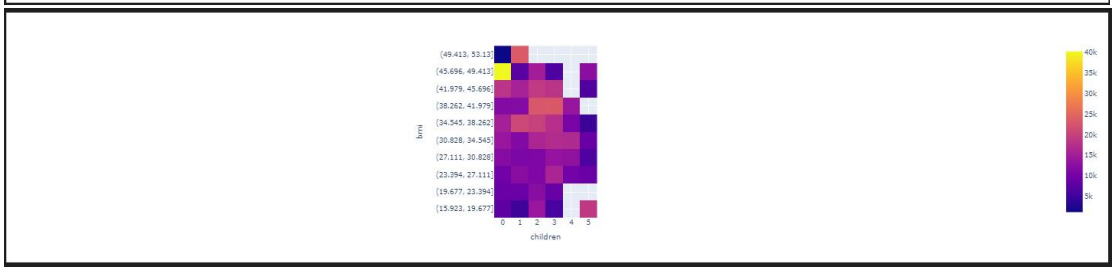
-Bản đồ nhiệt tương quan là bản đồ nhiệt mô tả ma trận tương quan hai chiều giữa hai chiều rời rạc, với các pixel màu biểu diễn dữ liệu trên thang màu. Các giá trị của chiều thứ nhất được hiển thị dưới dạng hàng trong bảng, trong khi các giá trị của chiều thứ hai xuất hiện dưới dạng cột. Sắc thái của ô tỷ lệ thuận với số phép đo tương ứng với giá trị chiều.
-Điều này làm cho bản đồ nhiệt tương quan trở nên tuyệt vời để phân tích dữ liệu vì chúng hiển thị sự khác biệt và phương sai trong cùng một dữ liệu trong khi vẫn tạo ra các mẫu có thể truy cập rõ ràng. Bản đồ nhiệt tương quan, giống như bản đồ nhiệt tiêu chuẩn, được hỗ trợ bởi thanh màu để làm cho dữ liệu dễ đọc và dễ hiểu hơn.

```
plot_heatmap(
    X[['age', 'bmi', 'children']],
    y,
    bins=10
)
projectpro.checkpoint('1e808c')
```

-**X[['age', 'bmi', 'children']]:** Phần này chọn lọc ra ba cột cụ thể ('age', 'bmi', 'children') từ DataFrame X chứa các đặc trưng. Nó tạo ra một DataFrame mới chỉ chứa dữ liệu từ ba cột này.
-**y**: Đại diện cho biến mục tiêu (biến bạn muốn dự đoán).
-**bins=10**: Đây là một tham số được truyền cho hàm **plot_heatmap**. Nó chỉ định rằng dữ liệu nên được chia thành 10 khoảng (**bins**) trước khi tính toán và hiển thị heatmap. Việc chia thành các khoảng giúp trực quan hóa dữ liệu liên tục một cách dễ hiểu hơn trên **heatmap**.
-**plot_heatmap(...):** Đây là một hàm tùy chỉnh, có thể được định nghĩa trong tệp **ml_pipeline/eda.py** (hoặc một vị trí tương tự). Nó nhận DataFrame chứa các đặc trưng đã chọn, biến mục tiêu và số lượng khoảng làm đầu vào. Hàm này chịu trách nhiệm tạo ra biểu đồ **heatmap**.
-**projectpro.checkpoint('1e808c'):** Dòng này là một điểm kiểm tra từ **ProjectPro**. Có thể nó được sử dụng để theo dõi tiến trình hoặc tạo điểm kiểm tra trong một dự án trên nền tảng **ProjectPro.**



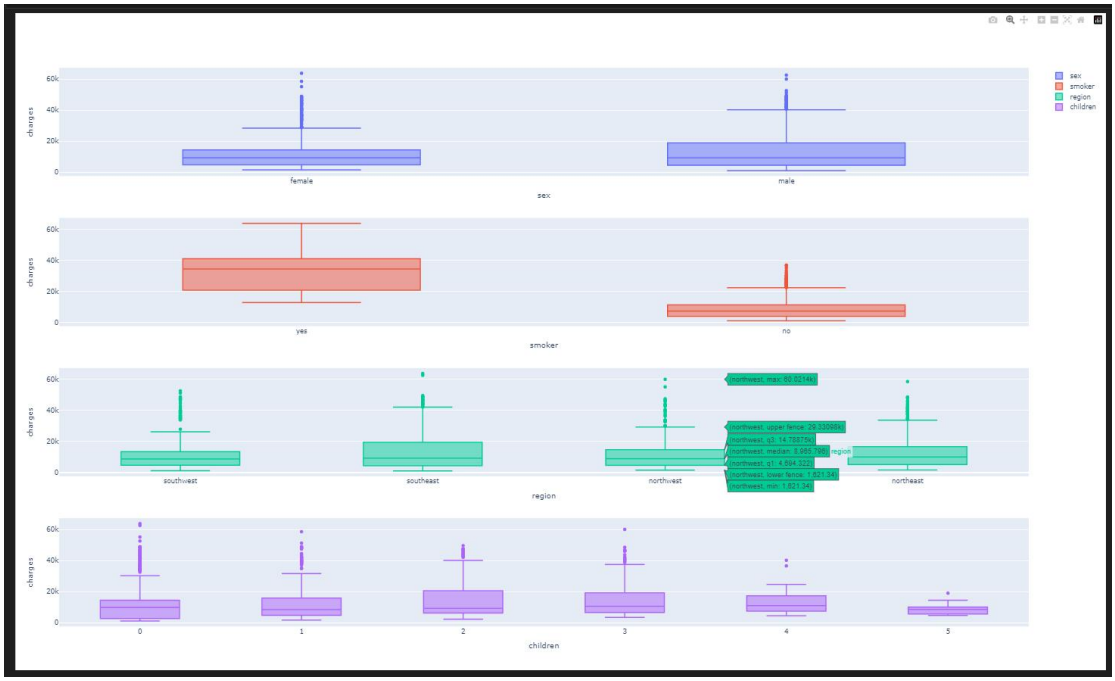




Cặp danh mục
Biểu đồ hộp

```
plot_paired_boxplots(
    X[['sex', 'smoker', 'region']],
    y
)
```

-X[['sex', 'smoker', 'region']]: Phần này chọn ra ba cột cụ thể (**'sex', 'smoker', 'region'**) từ DataFrame X chứa các đặc trưng của bạn. Các cột này được giả định là đại diện cho dữ liệu phân loại (ví dụ: '**sex**' có thể là 'nam' hoặc 'nữ', '**smoker**' có thể là 'có' hoặc 'không', và 'region' có thể có các tên vùng khác nhau).
-y: Đại diện cho biến mục tiêu, được kỳ vọng là một biến số (ví dụ: đại diện cho một giá trị liên tục như giá cả, chi phí hoặc một phép đo lường).
plot_paired_boxplots(...): Đây là một hàm tùy chỉnh, có thể được định nghĩa trong tệp **ml_pipeline/eda.py** (hoặc một vị trí tương tự). Nó nhận các đặc trưng phân loại và biến mục tiêu số làm đầu vào. Mục đích của hàm này là tạo ra các biểu đồ hộp được ghép nối.

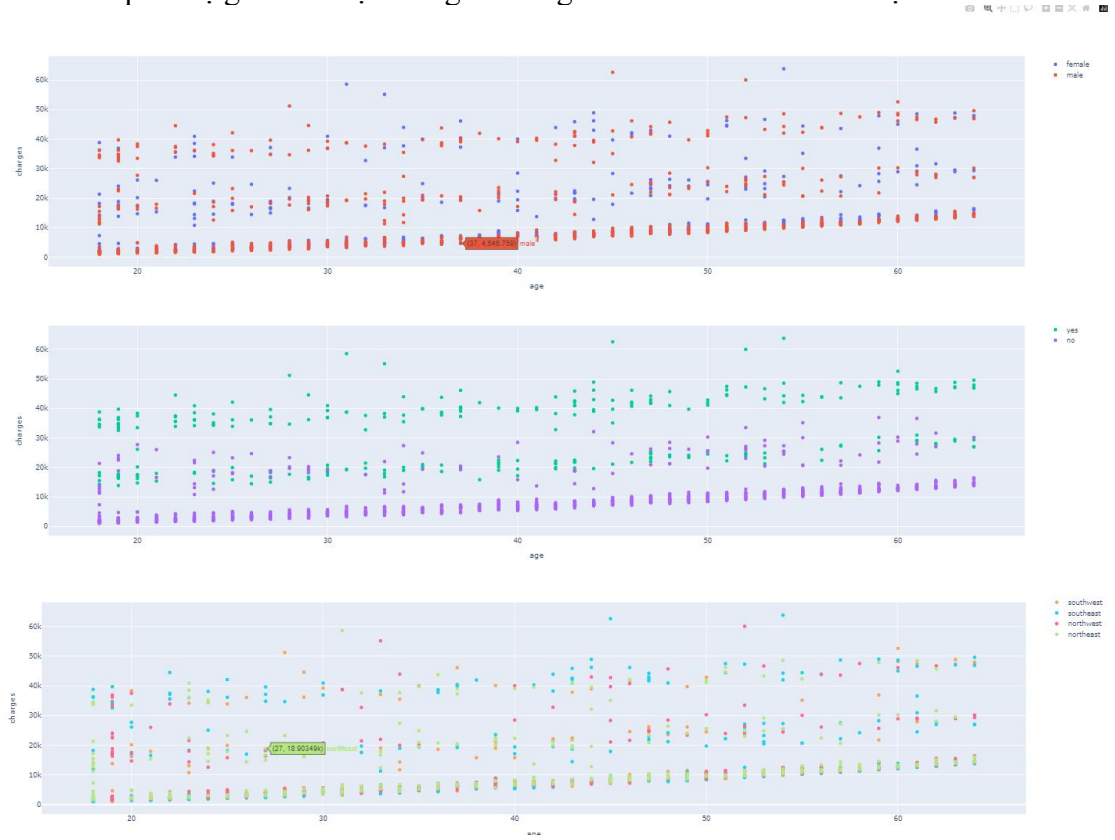


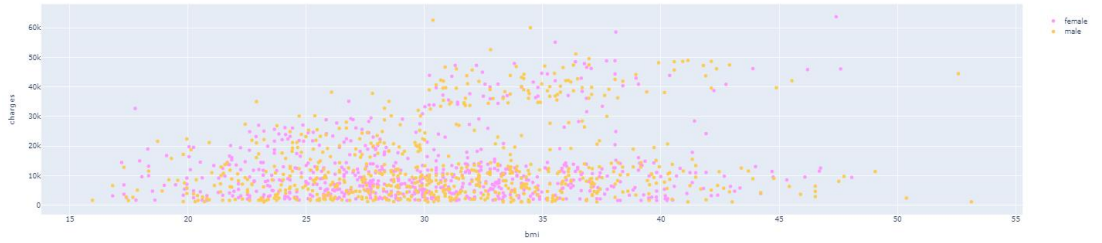
Những điểm cần lưu ý liên quan đến các cặp tính năng này:
- `**sex**`-`**smoker**` - `charges` trung bình cao hơn đối với nam giới hút thuốc so với nữ giới hút thuốc (36k so với 29k)
- `**smoker**`-`**region**` - `charges` trung bình cao hơn đối với người hút thuốc ở phía tây nam và đông nam so với đông bắc và tây bắc (35k và 37k so với 28k và 27k)

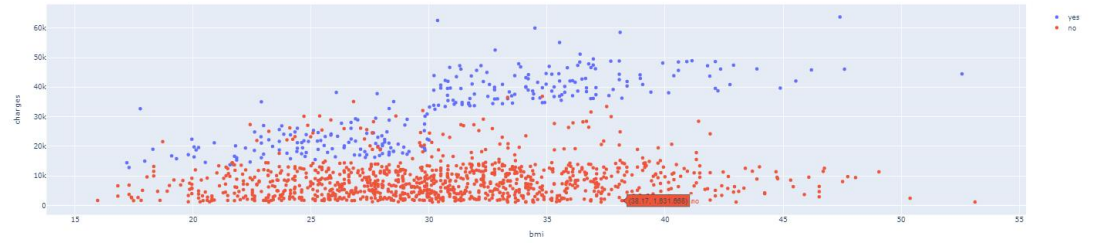
Cặp phân loại số

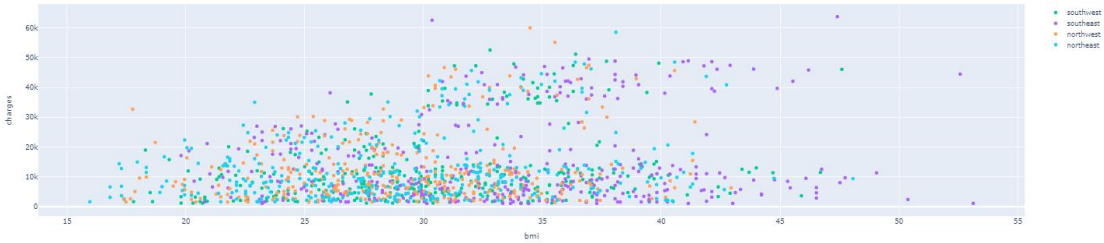
```
plot_paired_scatterplots(X, y)
```

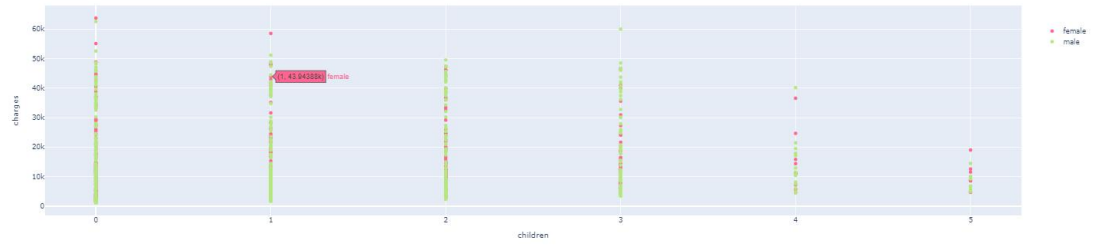
-Đoạn mã này có mục đích tạo ra các biểu đồ phân tán được ghép nối để trực quan hóa mối quan hệ giữa các đặc trưng số trong DataFrame X và biến mục tiêu số.

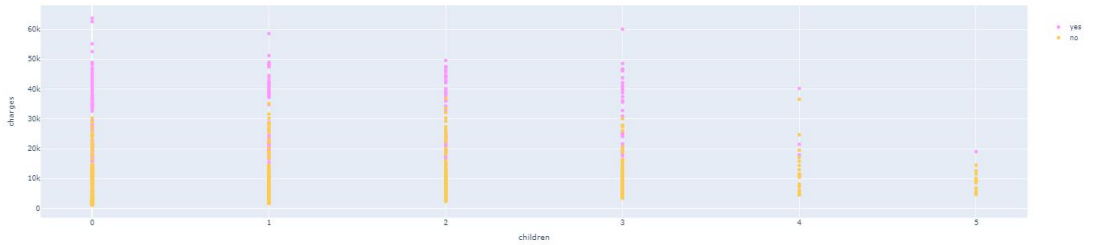


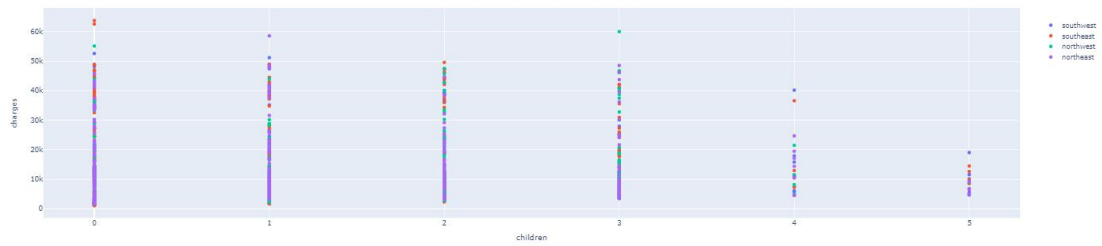












-Có hai thông tin chi tiết từ các biểu đồ này:
tuổi`-`người hút thuốc` - Có một nhóm ở góc dưới bên trái của biểu đồ (trong đó `tuổi` < 50 và `người hút thuốc` = 'không') trong đó tất cả những người thụ hưởng đều

có chi phí chăm sóc sức khỏe dưới 10 nghìn (tương đối nhỏ so với phần còn lại của dân số).
-`**bmi**`-`người hút thuốc` - Có một nhóm ở góc trên bên phải của biểu đồ (trong đó `bmi` > 30 và `người hút thuốc` = 'có') trong đó tất cả những người thụ hưởng đều có chi phí chăm sóc sức khỏe trên 30 nghìn (tương đối lớn so với phần còn lại của dân số). Có hai thông tin chi tiết từ các biểu đồ này:
- `tuổi`-`người hút thuốc` - Có một nhóm ở góc dưới bên trái của biểu đồ (trong đó `tuổi` < 50 và `người hút thuốc` = 'không') trong đó tất cả những người thụ hưởng đều có chi phí chăm sóc sức khỏe dưới 10 nghìn (tương đối nhỏ so với phần còn lại của dân số).
- `**bmi**`-`người hút thuốc` - Có một nhóm ở góc trên bên phải của biểu đồ (trong đó `**bmi**` > 30 và `người hút thuốc` = 'có') trong đó tất cả những người thụ hưởng đều có chi phí chăm sóc sức khỏe trên 30 nghìn (tương đối lớn so với phần còn lại của dân số).

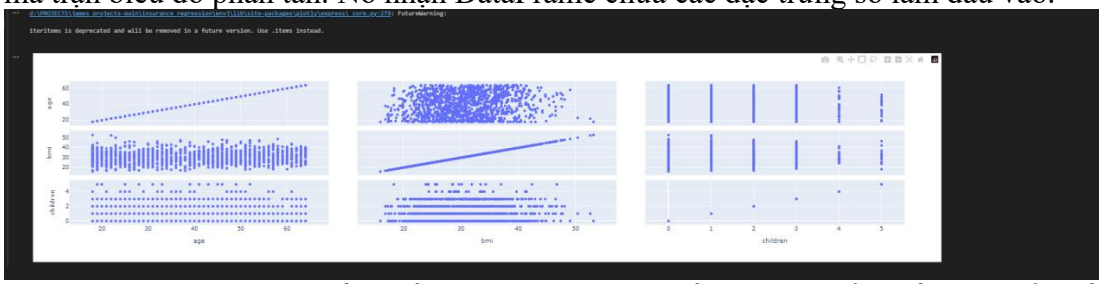
5.2.4.Cộng tuyến tính (giữa các đặc điểm)

Đặc điểm số

-Trước tiên, chúng ta hãy sử dụng biểu đồ cặp để xem từng cặp đặc điểm số liên quan đến nhau như thế nào:

```
px.scatter_matrix(
    X.select_dtypes(include=np.number)
```

-**X.select_dtypes(include=np.number**): Phần này của mã chọn lọc chỉ các đặc trưng số (các cột có kiểu dữ liệu số) từ DataFrame X. Nó sử dụng phương thức select_dtypes, cho phép bạn lọc các cột dựa trên kiểu dữ liệu của chúng. Bằng cách chỉ định include=np.number, nó chọn các cột có kiểu dữ liệu như số nguyên (**int64**) hoặc số thực (float64).
-**px.scatter_matrix(...):** Đây là lời gọi hàm chính từ thư viện **Plotly Express** để tạo ra ma trận biểu đồ phân tán. Nó nhận DataFrame chứa các đặc trưng số làm đầu vào.



-Có vẻ như không có nhiều mối tương quan giữa bất kỳ đặc điểm số nào. Để chắc chắn, chúng ta hãy tính toán và vẽ ma trận tương quan **Pearson**

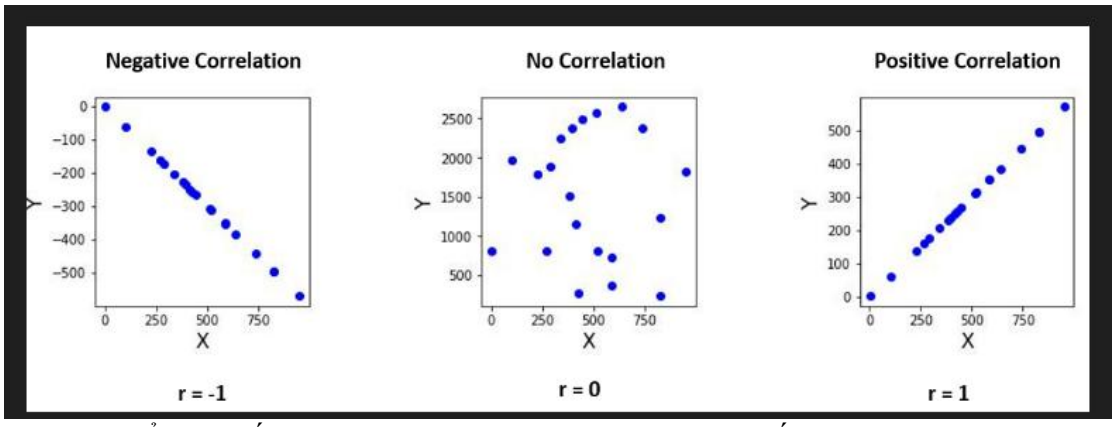
Tương quan

-Hệ số tương quan được sử dụng để đo cường độ mối quan hệ giữa hai biến. Hệ số này chỉ ra rằng khi giá trị của một biến thay đổi thì biến kia cũng thay đổi theo một hướng cụ thể với một số độ lớn. Có nhiều cách khác nhau để tìm mối tương quan giữa hai biến, một trong số đó là hệ số tương quan Pearson. Hệ số này đo mối quan hệ tuyến tính giữa hai biến liên tục.
-Giả sử x và y là hai biến liên tục, hệ số tương quan Pearson giữa chúng có thể được tìm thấy theo công thức sau.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

-trong đó **x_i** và **y_i** biểu diễn giá trị thứ **i^{th}** của các biến. Giá trị của r nằm trong khoảng từ -1 đến +1.
Mức độ liên hệ của chúng được đo bằng giá trị tuyệt đối của hệ số, trong khi dấu của hệ số biểu thị hướng của mối quan hệ.

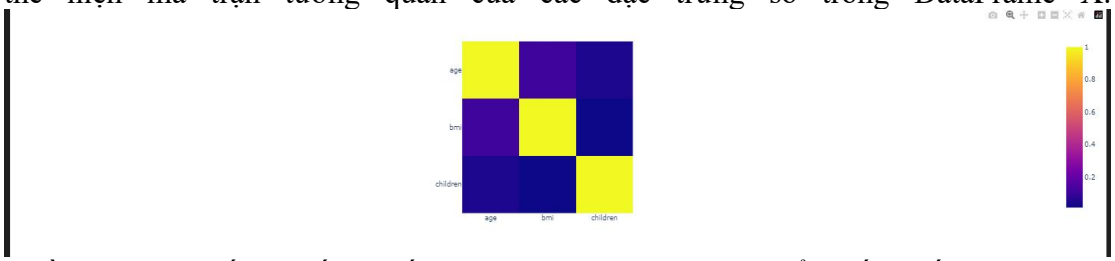
5.2.5.Đồ thị của các hệ số tương quan khác nhau



-**R = -1**: Biểu thị mối quan hệ tiêu cực hoàn toàn giữa các biế.
-**R =0**: Biểu thị không có mối quan hệ nào giữa các biến.
-**R= 1**: Biểu thị mối quan hệ tích cực hoàn toàn giữa các biến.

```
px.imshow(X.select_dtypes(include=np.number).corr())
```

-Đoạn mã này sử dụng Plotly Express (px) để tạo ra một biểu đồ heatmap (heatmap) thể hiện ma trận tương quan của các đặc trưng số trong DataFrame X.



-Điều này cho thấy có rất ít mối tương quan giữa các đặc điểm số - mối tương quan cao nhất là hệ số tương quan Pearson là 0,11.

Đặc điểm phân loại

Chi Squared test

- Phân phối của một biến phân loại trong mẫu thường phải được so sánh với phân phối của một biến phân loại trong một mẫu khác.
-Kiểm định chi bình phương về tính độc lập, thường được gọi là kiểm định chi bình phương về mối liên hệ, được sử dụng để phát hiện các biến phân loại có liên quan hay không.
-Dữ liệu phải đáp ứng các yêu cầu sau đối với kiểm định này:
* Biến phân loại
* Kích thước mẫu tương đối lớn

* Quan sát độc lập

-Kiểm định chi bình phương được thực hiện theo các bước sau:

+Tìm một số "mong đợi" (E) cho mỗi số quan sát (0) trong bảng.

$$\chi^2 = \frac{\sum(O_i - E_i)^2}{E_i}$$

Kiểm định chi bình phương sẽ cung cấp cho chúng ta giá trị p. Giá trị p cho biết kết quả kiểm định của chúng ta có ý nghĩa hay không.

-Tuy nhiên, để thực hiện kiểm định chi bình phương và tính giá trị p, chúng ta cần hai thông tin:

+Bậc tự do. Đơn giản là số lượng các danh mục trừ đi một.

+ Mức ý nghĩa. Mức alpha chuẩn là 0,05 (5%), nhưng có thể có các giá trị thay thế như 0,01 hoặc 0,10.

-Sau đây là các giá trị p khác nhau chỉ ra các cách giải thích giả thuyết khác nhau:

+**P = 0,05**; Giả thuyết bị bác bỏ

+ **P > 0,05**; Chấp nhận Giả thuyết

Watch the videos, Degrees of Freedom Part 1 and Degrees of Freedom Part 2 to understand the basics of Degrees of Freedom and their interpretation.

For categorical features, we'll use a Chi Squared (χ^2) test to observe whether each categorical feature pair is correlated.

-Đầu tiên, chúng ta tính toán các giá trị x^2, giá trị p và bậc tự do:

```
X_chi2 = chi2(X.select_dtypes(object))
```

-Đoạn mã này thực hiện kiểm định chi bình phương (chi-squared test) trên các đặc trưng phân loại (categorical features) trong DataFrame X và lưu trữ kết quả vào biến X_chi2

```
X_chi2
```

-Đoạn mã này là truy cập và hiển thị (hoặc sử dụng) biến X_chi2 đã được tính toán trước đó.

	column1	column2	chi_squared	p_value	dof
0	sex	smoker	7.392911	0.006548	1
1	sex	region	0.435137	0.932892	3
2	smoker	region	7.343478	0.061720	3

Vì chỉ có ba cặp đặc điểm phân loại khác nhau, nên chúng tôi sẽ không vẽ biểu đồ kết quả.

- Chúng tôi có thể sử dụng giá trị p (được suy ra từ kết quả x^2 và các bậc tự do) để kiểm tra khả năng phân phối quan sát được (cho mỗi cặp đặc điểm) xảy ra do ngẫu nhiên. Ví dụ: giá trị p bằng 0,5 có nghĩa là có 50% khả năng quan sát phân phối ngẫu nhiên.

-Ngưỡng < 0,05 được chấp nhận rộng rãi để bác bỏ giả thuyết vô hiệu (rằng các đặc điểm độc lập) vì điều này có nghĩa là chỉ có 5% khả năng quan sát phân phối ngẫu nhiên.

-Cặp đặc điểm duy nhất có giá trị p nhỏ hơn ngưỡng này là `**sex**` và `**smoker**`, điều này có nghĩa là có khả năng các đặc điểm này có tương quan:

```
X_chi2[X_chi2['p_value'] < 0.05]
```

-Đoạn mã này lọc kết quả của kiểm định chi bình phương (chi-squared test) được lưu trữ trong biến **X_chi2** để chỉ chọn ra những đặc trưng có giá trị **p (p-value)** nhỏ hơn 0.05.

	column1	column2	chi_squared	p_value	dof
0	sex	smoker	7.392911	0.006548	1

Cặp tính năng số-phân loại

ANOVA

-Đối với cặp tính năng số-phân loại, chúng ta sẽ sử dụng thử nghiệm ANOVA. ANOVA là viết tắt của Phân tích phương sai - giúp chúng ta hiểu liệu có sự khác biệt đáng kể về mặt thống kê giữa các phương tiện của các nhóm độc lập hay không.

-ANOVA là viết tắt của Phân tích phương sai, đánh giá sự khác biệt giữa các phương tiện của nhóm. Đây là một thử nghiệm giả thuyết thống kê nhằm xem liệu phương tiện của ít nhất hai quần thể có khác nhau hay không. Để chạy ANOVA, chúng ta cần ít nhất một biến liên tục và một biến phân loại để phân tách dữ liệu của bạn thành các nhóm so sánh. Thuật ngữ "phân tích phương sai" đề cập đến cách thử nghiệm sử dụng phương sai để xác định xem các phương tiện có khác nhau hay không.

-ANOVA so sánh phương sai của phương tiện của nhóm với phương sai của các nhóm. Quy trình này xác định xem các nhóm có phải là một phần của quần thể lớn hơn hay chúng là các quần thể riêng biệt có các phương tiện khác nhau.

-Mặc dù thực tế là nó phân tích các biến thể, nhưng nó kiểm tra các phương tiện. Một cách ANOVA là loại ANOVA cơ bản nhất. Phương pháp này là tổng quát hóa các bài kiểm tra t có thể được sử dụng để so sánh nhiều hơn hai nhóm.

-Giả thuyết không là tất cả các nhóm có cùng giá trị trung bình và giả thuyết thay thế là ít nhất một nhóm có giá trị trung bình khác nhau.

- Đầu tiên, chúng tôi tính toán các giá trị F (tỷ lệ giữa các bình phương trung bình giữa và trong nhóm) và các giá trị p:

```
X_anova = anova(X)
```

-Đoạn mã này thực hiện phân tích phương sai (ANOVA) trên DataFrame X và lưu trữ kết quả vào biến **X_anova**.

```
X_anova
```

-Đoạn mã này là truy cập và hiển thị (hoặc sử dụng) biến **X_anova** đã được tính toán trước đó.

	num_column	cat_column	f_stat	p_value
0	age	sex	0.581369	4.459107e-01
1	age	smoker	0.836777	3.604853e-01
2	age	region	0.079782	9.709891e-01
3	bmi	sex	2.878970	8.997637e-02
4	bmi	smoker	0.018792	8.909850e-01
5	bmi	region	39.495057	1.881839e-24
6	children	sex	0.393659	5.304898e-01
7	children	smoker	0.078664	7.791596e-01
8	children	region	0.717493	5.415543e-01

-Tương tự như phép kiểm định x^2 ở trên, chúng ta có thể sử dụng giá trị p để kiểm tra khả năng phân phối quan sát được (cho mỗi cặp tính năng) xảy ra do ngẫu nhiên. Chúng ta sẽ lại áp dụng ngưỡng < 0,05 để bác bỏ giả thuyết không (rằng các tính năng là độc lập).

-Cặp tính năng duy nhất có giá trị p nhỏ hơn ngưỡng này là `**bmi**` và `**region**`, điều này có nghĩa là có khả năng những tính năng này có mối tương quan:

```
X_anova[X_anova['p_value'] < 0.05]
```

	num_column	cat_column	f_stat	p_value
5	bmi	region	39.495057	1.881839e-24

-Đoạn mã này lọc kết quả của kiểm định ANOVA được lưu trữ trong biến **X_anova** để chỉ chọn ra những đặc trưng có giá trị p (p-value) nhỏ hơn 0.05.

5.2.6.Tương quan (so với mục tiêu)]

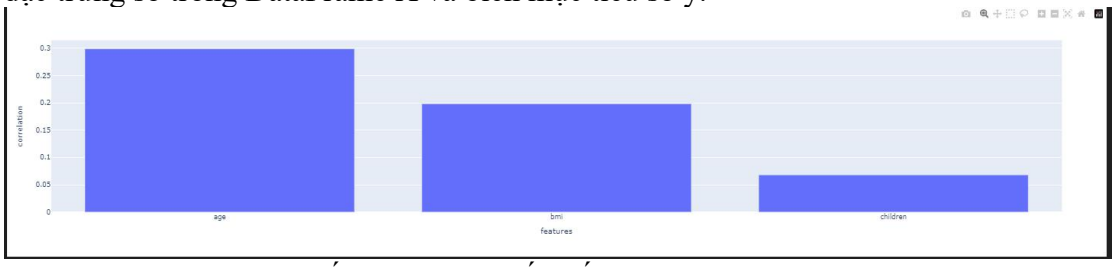
-Bây giờ chúng ta sẽ tính toán tương quan giữa các đặc điểm và mục tiêu. Điều này sẽ hữu ích khi xây dựng mô hình tuyến tính cơ sở của chúng ta.
-Chúng ta có thể áp dụng các kỹ thuật tương tự mà chúng ta đã sử dụng khi tính toán tính cộng tuyến giữa các đặc điểm.

Các đặc điểm số (so với mục tiêu)

-Tại đây, chúng ta sẽ tính toán Tương quan Pearson giữa từng đặc điểm số và mục tiêu:

```
plot_pearson_wrt_target(X, y)
```

-Đoạn mã này có mục đích tạo ra biểu đồ thể hiện mối tương quan Pearson giữa các đặc trưng số trong DataFrame X và biến mục tiêu số y.



-Tính năng `trẻ em` có mối tương quan rất thấp so với mục tiêu.

Các đặc điểm phân loại (liên quan đến mục tiêu).
-Ở đây, chúng tôi sẽ áp dụng thử nghiệm ANOVA. Lưu ý rằng vì chúng tôi chỉ quan tâm đến việc so sánh từng đặc điểm phân loại với mục tiêu, nên chúng tôi lọc ra kết quả của tất cả các đặc điểm số khác:

```
data_anova = anova(data) # Use data as it contains the target
projectpro.checkpoint('1e808c')
anova_wrt_target = data_anova[data_anova['num_column']=='charges']
```

-Đoạn mã này thực hiện phân tích phương sai (ANOVA) trên toàn bộ DataFrame data (bao gồm cả biến mục tiêu) và sau đó lọc kết quả để chỉ lấy những thông tin liên quan đến biến mục tiêu 'charges'.

```
anova_wrt_target
```

-Đoạn mã này đơn giản là truy cập và hiển thị (hoặc sử dụng) biến anova_wrt_target đã được tính toán và lọc trước đó.

	num_column	cat_column	f_stat	p_value
9	charges	sex	4.399702	3.613272e-02
10	charges	smoker	2177.614868	8.271436e-283
11	charges	region	2.969627	3.089336e-02

-Tất cả các giá trị p đều < 0,05, nghĩa là sự khác biệt được quan sát thấy trong cột `charges` khi so sánh các danh mục trong một biến phân loại là có ý nghĩa thống kê. Tuy nhiên, lưu ý rằng điều này không đo lường được mức độ khác biệt được quan sát thấy.

5.3 Xây dựng mô hình đánh giá cơ sở

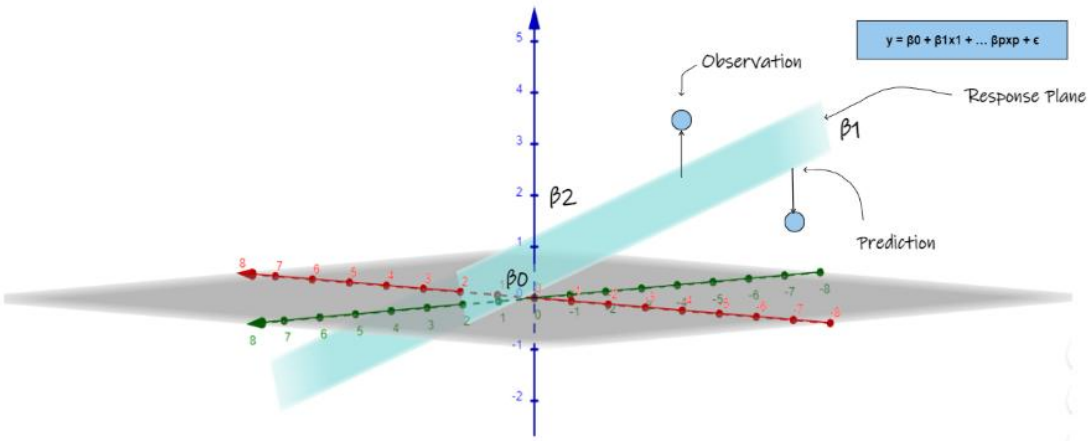
5.3.1 Hồi quy tuyến tính đa biến

Hồi quy tuyến tính, thường được gọi là hồi quy đơn giản, tạo ra mối quan hệ giữa hai biến. Hồi quy tuyến tính được biểu diễn trực quan dưới dạng một đường thẳng, với độ dốc xác định cách thay đổi của một biến ảnh hưởng đến thay đổi của biến kia. Giao điểm y của một mối quan hệ hồi quy tuyến tính phản ánh giá trị của một biến khi giá trị của biến kia bằng 0.
Hồi quy tuyến tính đa biến ước tính mối quan hệ giữa hai hoặc nhiều biến độc lập và một biến phụ thuộc.

$$\hat{y} = \beta_0 + \beta_1 + \ldots + \beta_p x_p + \epsilon$$

trong đó *p* là... số lượng đặc trưng trong mô hình.

- Đối với bất kỳ giá trị biến độc lập (x) nào, y là giá trị dự đoán của biến phụ thuộc (y).
- β_0 biểu thị hệ số chặn, hoặc giá trị dự đoán của y, khi x bằng 0.
- β_1 là hệ số hồi quy của biến x1, cho biết y sẽ thay đổi bao nhiêu khi x1 tăng hoặc giảm.
- β_p là hệ số hồi quy của biến cuối cùng xp, cho biết y sẽ thay đổi bao nhiêu khi xp tăng hoặc giảm.
- x_1 ... x_p là các biến độc lập hoặc biến dự đoán giúp chúng ta dự đoán y
- ϵ là sai số còn lại



5.3.2 Các giả định của hồi quy tuyến tính

Một mô hình hồi quy tuyến tính có các giả định sau:

1. Tính độc lập của các quan sát: Các quan sát khác nhau không liên quan đến nhau.
2. Tuyến tính: Mối quan hệ tuyến tính giữa biến mục tiêu và các đặc trưng.
3. Tính độc lập: Ít hoặc không có đa cộng tuyến giữa các đặc trưng.

4. Tính chuẩn của phần dư: Phần dư được phân phối chuẩn.
5. Đồng phương sai: Phần dư có phương sai giống nhau trên các giá trị của biến mục tiêu.

*Lưu ý rằng:

- Giả định 1 đúng với tập dữ liệu này, vì chi phí y tế của một người thụ hưởng không nên liên quan đến người khác.
- Chúng ta sẽ sử dụng giả định 2-3 để định hướng cách chúng ta xử lý dữ liệu.
- Chúng ta sẽ sử dụng giả định 4-5 để kiểm tra xem mô hình tuyến tính của chúng ta có phù hợp với mục đích hay không.

Sai số trong hồi quy:
Đường hồi quy tiến về phía giá trị trung bình để tạo ra sự phù hợp tốt nhất, về cơ bản có nghĩa là sai số ở mức thấp nhất. Trong biểu đồ trên, có thể thấy rằng đường hồi quy không thể dự đoán chính xác các giá trị thực. Luôn luôn có một khoảng trống cho sai số.
Hãy cùng tìm hiểu các loại sai số khác nhau trong Hồi quy:

- Sai số tuyệt đối trung bình (MAE) là thống kê sai số hồi quy cơ bản nhất để nắm bắt. Chúng ta sẽ tính toán phần dư cho mỗi điểm dữ liệu riêng lẻ, chỉ sử dụng giá trị tuyệt đối của mỗi điểm để các phần dư âm và dương không triệt tiêu lẫn nhau. Sau đó, giá trị trung bình của tất cả các phần dư này được tính toán. MAE về cơ bản mô tả độ lớn điển hình của các phần dư.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y - \hat{y}|$$

- Sai số bình phương trung bình (MSE) giống với sai số tuyệt đối trung bình (MAE) nhưng bình phương hiệu số trước khi tổng hợp tất cả chúng. MSE gần như luôn lớn hơn MAE vì chúng ta đang bình phương hiệu số. Do đó, chúng ta không thể so sánh trực tiếp MAE và MSE.Chúng ta bị giới hạn trong việc so sánh các chỉ số sai số của mô hình của chúng ta với các chỉ số của mô hình đối thủ. Sự hiện diện của các giá trị ngoại lai trong dữ liệu của chúng ta làm cho tác động của thuật ngữ bình phương lên phương trình MSE rất rõ ràng. Trong MAE, mỗi phần dư được cộng tỷ lệ thuận với tổng sai số, trong khi trong MSE, sai số tăng theo cấp số nhân. Kết quả là, các giá trị ngoại lai trong dữ liệu của chúng ta cuối cùng sẽ dẫn đến tổng sai số lớn hơn đáng kể trong MSE so với trong MAE. Tương tự như vậy, mô hình của chúng ta sẽ bị ảnh hưởng nhiều hơn nếu nó dự đoán các giá trị khác biệt đáng kể so với giá trị thực tế tương ứng. Điều này có nghĩa là trong MSE, trái ngược với MAE, những chênh lệch đáng kể giữa giá trị thực tế và giá trị dự đoán bị phạt nặng hơn.
- Nếu chúng ta muốn hạn chế tầm quan trọng của các giá trị ngoại lai, chúng ta nên sử dụng MAE vì các phần dư ngoại lai không đóng góp nhiều vào tổng sai số như MSE. Cuối cùng, quyết định giữa MSE và MAE phụ thuộc vào ứng dụng cụ thể và cách xử lý các sai số lớn.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y - \hat{y})^2$$

- Sai số bình phương trung bình (RMSE) là một thống kê sai số khác mà bạn có thể gặp phải. Nó là căn bậc hai của MSE, như tên gọi của nó. Vì MSE được bình phương, nên đơn vị của nó khác với đầu ra ban đầu. RMSE thường được sử dụng để chuyển đổi chỉ số sai số trở lại đơn vị có thể so sánh được, giúp việc diễn giải dễ dàng hơn. Các giá trị ngoại lai có tác động tương đương đối với MSE và RMSE vì cả hai đều bình phương phần dư.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2}$$

- Đối tác phần trăm của MAE là sai số phần trăm tuyệt đối trung bình (MAPE). Giống như MAE là lượng sai số trung bình do mô hình của bạn tạo ra, MAPE là khoảng cách trung bình giữa các dự đoán của mô hình và đầu ra tương ứng của chúng. MAPE, giống như MAE, có ý nghĩa rõ ràng vì phần trăm dễ hiểu hơn đối với mọi người. Do sử dụng giá trị tuyệt đối, cả MAPE và MAE đều có khả năng kháng lại ảnh hưởng của các giá trị ngoại lai.

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y - \hat{y}}{y}\right|$$

5.3.3 So sánh mô hình

R^2: R^2 hoặc hệ số xác định là tỷ lệ phương sai của biến phụ thuộc được giải thích từ (các) biến độc lập. R^2 được biểu thị trong khoảng từ 0 đến 1 cho mức độ phương sai được giải thích. Như chúng ta đã học trong phần trước, tỷ lệ $\frac{SSE}{TSS}$ nên thấp đối với một mô hình mạnh mẽ, tỷ lệ này biểu thị sai số hoặc phương sai không được giải thích bởi (các) biến độc lập. Về mặt toán học, R^2 hoặc phương sai được giải thích có thể được biểu thị như sau:

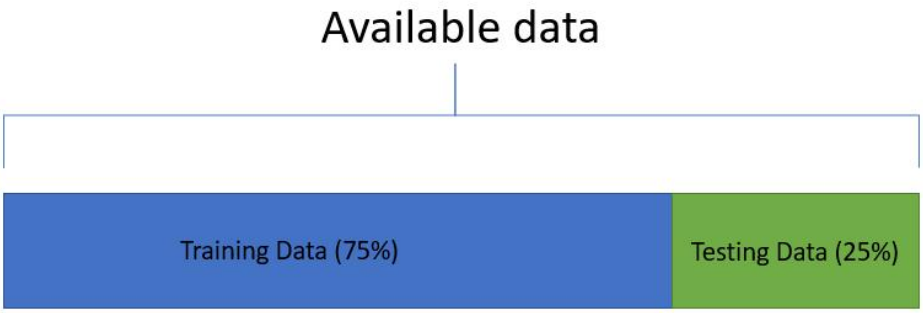
$$R2 = 1 - \frac{SSE}{TSS}$$

Hiệu chỉnh R^2: Đối với các mô hình tuyến tính, R^2 hiệu chỉnh là một thống kê đo lường độ phù hợp đã được hiệu chỉnh. Nó xác định tỷ lệ phương sai trong trường mục tiêu được giải thích bởi đầu vào hoặc các đầu vào. R^2 có xu hướng đánh giá quá cao độ phù hợp của hồi quy tuyến tính. Nó luôn tăng khi số lượng biến độc lập trong mô hình tăng. Điều này xảy ra vì chúng ta có xu hướng khấu trừ một lượng lớn (do nhiều biến) để tính toán sai số khi số lượng biến độc lập tăng lên. Do đó, tỷ lệ $\frac{SSE}{TSS}$ thậm chí còn thấp hơn mức cần thiết và R^2 có vẻ cao nhưng nó có thể không phải là một mô hình phù hợp cho dữ liệu thực tế. Nó được điều chỉnh để tính đến sự đánh giá quá cao này. Xét N là tổng kích thước mẫu và p là số lượng biến độc lập, R^2 hiệu chỉnh có thể được biểu thị như sau:

$$\text{Hiệu chỉnh } R^2 = 1 - \frac{(1-R^2)(N-1)}{N-p-1}$$

5.3.4 Tiền xử lý dữ liệu

Chia dữ liệu thành huấn luyện và kiểm tra



Trong phương pháp chia dữ liệu thành tập huấn luyện và tập kiểm tra, các điểm dữ liệu được chia thành hai tập dữ liệu: tập huấn luyện và tập kiểm tra. Dữ liệu huấn luyện được sử dụng để huấn luyện mô hình, và sau đó mô hình được sử dụng để dự đoán trên dữ liệu kiểm tra để xem mô hình hoạt động như thế nào trên dữ liệu chưa nhìn thấy và liệu nó có bị quá khớp (overfitting) hoặc thiếu khớp (underfitting) hay không.

Luôn luôn là thực hành tốt nhất để chia tập dữ liệu của bạn thành tập huấn luyện (được sử dụng để huấn luyện bất kỳ bước xử lý dữ liệu nào và bản thân mô hình) và tập kiểm tra (chỉ được sử dụng để đánh giá mô hình).

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(
        X,
        y,
        test_size=0.33,
        random_state=42
    )
```

5.3.5 Loại bỏ các cột không cần thiết

Sau khi thực hiện Phân tích Khám phá Dữ liệu (EDA), chúng tôi kết luận rằng đặc trưng "children" (số con) không có tương quan mạnh với "charges" (chi phí). Điều này vi phạm Giả định 2 của mô hình hồi quy tuyến tính, vì vậy chúng ta nên loại bỏ đặc trưng này khỏi tập huấn luyện.

Ngoài ra, kiểm định x^2 của chúng tôi cho thấy đặc trưng "sex" (giới tính) có tương quan với đặc trưng "smoker" (người hút thuốc), và kiểm định ANOVA của chúng tôi cho thấy đặc trưng "region" (khu vực) có tương quan với đặc trưng "bmi" (chỉ số khối cơ thể). Điều này vi phạm Giả định 3 của mô hình hồi quy tuyến tính, vì vậy chúng ta nên loại bỏ các đặc trưng này khỏi tập huấn luyện. Chúng tôi sẽ loại bỏ các đặc trưng "sex" và "region", vì chúng có sức mạnh dự đoán yếu hơn đối với biến mục tiêu.

```
cols_to_drop = [
    'children',
    'region',
    'sex'
]
X_train.drop(cols_to_drop, axis=1, inplace=True)
X_test.drop(cols_to_drop, axis=1, inplace=True)
```

5.3.6 Mã hóa

Mã hóa One-Hot

Hầu hết các thuật toán Học máy không thể xử lý dữ liệu phân loại và phải được chuyển đổi thành dữ liệu số. Các lựa chọn của chúng ta để chuyển đổi dữ liệu phân loại thành dữ liệu số là gì? Chúng ta có nên đánh số các nhãn không? Ví dụ, đỏ biểu

thị 0 và xanh lam biểu thị 1. Điều này sẽ gây ra sai lệch trong mô hình vì mô hình sẽ coi xanh lam là cao hơn đỏ.
Để giải quyết vấn đề này, chúng ta sử dụng một phương pháp mã hóa dữ liệu phân loại được gọi là mã hóa one-hot. Trong phương pháp này, chúng ta tạo một đặc trưng mới cho mỗi nhãn và gán cho nó giá trị là 1. Ví dụ, nếu màu xanh lam có mặt, nó được đánh dấu là 1, ngược lại là 0.
Bây giờ, chúng ta cần mã hóa các đặc trưng phân loại của mình - điều này có nghĩa là biến đổi chúng thành các đặc trưng số để mô hình có thể diễn giải chúng.
Có nhiều phương pháp mã hóa, nhưng trong ví dụ này, chúng ta sẽ sử dụng Mã hóa One-Hot, phương pháp tạo một cột boolean cho mỗi danh mục trong mỗi đặc trưng phân loại.
Lưu ý rằng chúng ta sẽ sử dụng fit và transform trên tập huấn luyện và chỉ sử dụng transform trên tập kiểm tra, điều này về cơ bản có nghĩa là đối tượng mã hóa one-hot được huấn luyện hoặc khớp bằng cách chỉ xem các giá trị của tập huấn luyện.

```
ohe = OneHotEncoder(use_cat_names=True)
X_train = ohe.fit_transform(X_train)
X_test = ohe.transform(X_test)
projectpro.checkpoint('1e808c')
```

- **ohe = OneHotEncoder(use_cat_names=True):**
  - Dòng này tạo ra một đối tượng OneHotEncoder và gán nó cho biến ohe.
  - use_cat_names=True yêu cầu bộ mã hóa sử dụng tên danh mục gốc làm tên đặc trưng trong đầu ra đã được mã hóa. Điều này giúp cải thiện khả năng diễn giải.
- **X_train = ohe.fit_transform(X_train):**

  Dòng này thực hiện hai việc:
  - **fit:** Bộ mã hóa học các danh mục duy nhất hiện có trong tập dữ liệu **X_train**. Nó xác định tất cả các giá trị riêng biệt trong các đặc trưng phân loại.
  - **transform**: Dữ liệu **X_train** được biến đổi bằng cách sử dụng các danh mục đã học. Thay thế các giá trị phân loại bằng các cột được mã hóa one-hot.
  - Kết quả (tập **X_train** đã được biến đổi) được gán lại cho biến **X_train**, ghi đè lên dữ liệu gốc.
- **X_test = ohe.transform(X_test):** Dòng này chỉ biến đổi dữ liệu **X_test** bằng cách sử dụng các danh mục mà bộ mã hóa đã học từ **X_train**.
  - Điều quan trọng là chỉ sử dụng transform ở đây để ngăn chặn rò rỉ dữ liệu (sử dụng thông tin từ tập kiểm tra trong quá trình huấn luyện).
  - Kết quả được gán lại cho **X_test**.
- **projectpro.checkpoint('1e808c'):** Dòng này có thể là một lệnh gọi hàm dành riêng cho thư viện **projectpro** hoặc nền tảng bạn đang sử dụng.
  - Nó dường như đang tạo ra một điểm kiểm tra hoặc lưu tiến trình với một mã định danh ('**1e808c**').

Vì đặc trưng **"smoker"** (người hút thuốc) chỉ có hai danh mục, chúng ta có thể loại bỏ cột được mã hóa **"smoker_no"** (không hút thuốc), vì nó không cung cấp thêm bất kỳ thông tin nào:

```
cols_to_drop = ['smoker_no']
X_train.drop(cols_to_drop, axis=1, inplace=True)
X_test.drop(cols_to_drop, axis=1, inplace=True)
```

5.3.7 Biến đổi biến mục tiêu

Trong phần EDA, chúng ta đã thấy rằng biến mục tiêu không phân phối chuẩn. Một mô hình được huấn luyện bằng cách sử dụng biến mục tiêu này có thể sẽ tạo ra các phần dư không có phương sai giống nhau trên các giá trị của biến mục tiêu (tức là giả định 5 của mô hình hồi quy tuyến tính sẽ bị vi phạm).
Các phép biến đổi lũy thừa là một họ các phép biến đổi tham số, đơn điệu được áp dụng để làm cho dữ liệu giống với phân phối Gaussian hơn. Điều này hữu ích cho các vấn đề mô hình hóa liên quan đến tính không đồng nhất phương sai (phương sai không đổi), hoặc các tình huống khác khi mong muốn có tính chuẩn.
Hiện tại, PowerTransformer hỗ trợ phép biến đổi Box-Cox và phép biến đổi Yeo-Johnson.
Vì vậy, chúng ta sẽ sử dụng phép biến đổi Yeo-Johnson để đảm bảo biến mục tiêu của chúng ta phân phối chuẩn hơn:

```
pt = PowerTransformer(method='yeo-johnson')
y_train_t = pt.fit_transform(y_train.values.reshape(-1, 1))[:, 0]
y_test_t = pt.transform(y_test.values.reshape(-1, 1))[:, 0]
```

pt = PowerTransformer(method='yeo-johnson'):

- Dòng này tạo ra một đối tượng PowerTransformer và gán nó cho biến pt.
- method='yeo-johnson' chỉ định rằng phép biến đổi Yeo-Johnson sẽ được sử dụng. Phép biến đổi này được chọn vì nó có thể xử lý cả dữ liệu có giá trị âm và giá trị dương.

y_train_t = pt.fit_transform(y_train.values.reshape(-1, 1))[:, 0]:

- Dòng này thực hiện hai việc:
  - **fit:** Bộ biến đổi pt được khớp với dữ liệu huấn luyện **y_train**. Điều này có nghĩa là nó sẽ tính toán các tham số cần thiết để thực hiện phép biến đổi Yeo-Johnson.
  - **transform:** Dữ liệu huấn luyện **y_train** được biến đổi bằng cách sử dụng phép biến đổi Yeo-Johnson với các tham số đã được tính toán.
- **y_train.values.reshape(-1, 1):** Thao tác này được thực hiện để đảm bảo rằng dữ liệu y_train có dạng phù hợp với PowerTransformer. Nó chuyển đổi dữ liệu thành một mảng 2 chiều với một cột.
- **[:, 0]** : Lấy cột đầu tiên (và duy nhất) từ kết quả của **fit_transform**, đảm bảo **y_train_t** là một mảng 1 chiều.
- Kết quả (dữ liệu huấn luyện đã được biến đổi) được gán cho biến **y_train_t**.

y_test_t = pt.transform(y_test.values.reshape(-1, 1))[:, 0]:

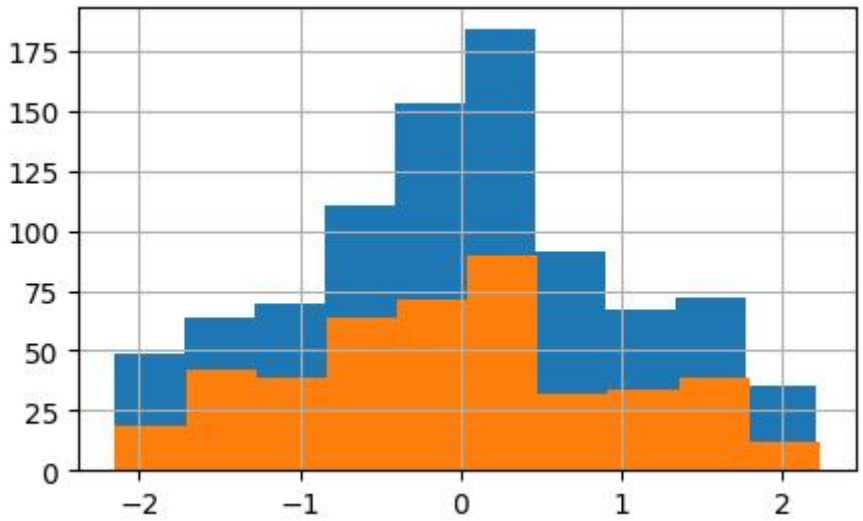
- Dòng này chỉ biến đổi dữ liệu kiểm tra **y_test** bằng cách sử dụng phép biến đổi Yeo-Johnson với các tham số đã được tính toán từ dữ liệu huấn luyện.
- Nó không khớp lại bộ biến đổi với dữ liệu kiểm tra để tránh rò rỉ dữ liệu.
- Tương tự như trên, **y_test.values.reshape(-1, 1)** và **[:, 0]** được sử dụng để định dạng dữ liệu và lấy kết quả mong muốn.

- Kết quả (dữ liệu kiểm tra đã được biến đổi) được gán cho biến **y_test_t.**

Bây giờ, hãy nhanh chóng kiểm tra xem điều này có mang lại hiệu quả mong muốn hay không (trên cả tập huấn luyện và tập kiểm tra):

```
pd.Series(y_train_t).hist(figsize=(5, 3))
pd.Series(y_test_t).hist(figsize=(5, 3))
```

- **pd.Series(y_train_t):** Tạo một Series Pandas từ dữ liệu y_train_t (biến mục tiêu đã được biến đổi trên tập huấn luyện).
- **.hist(figsize=(5, 3)):** Vẽ biểu đồ histogram cho Series vừa tạo.
- figsize=(5, 3): Thiết lập kích thước của biểu đồ (chiều rộng 5, chiều cao 3).
- **pd.Series(y_test_t).hist(figsize=(5, 3)):** Làm tương tự như trên, nhưng với dữ liệu y_test_t (biến mục tiêu đã được biến đổi trên tập kiểm tra)



5.3.8 Huấn luyện mô hình

Bây giờ, khi tập dữ liệu của chúng ta đã được xử lý đầy đủ, chúng ta có thể huấn luyện mô hình hồi quy tuyến tính cơ sở của mình.

Để cải thiện tính đồng nhất phương sai của phần dư, chúng ta có thể truyền trọng số mẫu cho mô hình. Điều này đảm bảo rằng khi mô hình được huấn luyện, các quan sát có chi phí lớn hơn sẽ được coi trọng hơn so với các quan sát có chi phí nhỏ hơn. Điều này có nghĩa là phần dư sẽ bị phạt nặng hơn đối với các quan sát có chi phí lớn hơn so với các quan sát có chi phí nhỏ hơn.

Chúng ta sẽ sử dụng cột mục tiêu làm trọng số mẫu, nhưng sẽ chia tỷ lệ nó theo giá trị tối thiểu của cột chi phí (vì vậy trọng số mẫu tối thiểu là 1

```
sample_weight = y_train / y_train.min()
```

- **y_train:** Đây là biến chứa giá trị của biến mục tiêu (ví dụ: chi phí) cho tập huấn luyện.
- **y_train.min():** Lấy giá trị tối thiểu của **y_train**.
- **/:** Thực hiện phép chia từng phần tử của **y_train** cho giá trị tối thiểu của **y_train**.

Cuối cùng, chúng ta có thể huấn luyện mô hình hồi quy tuyến tính của mình bằng cách truyền cho nó tập huấn luyện và trọng số mẫu:

```
lr = LinearRegression()
lr.fit(
    X_train,
    y_train_t,
    sample_weight=sample_weight
)
```

```
▾ LinearRegression
LinearRegression()
```

5.3.9 Đánh giá mô hình

Bây giờ, khi đã huấn luyện xong mô hình, chúng ta có thể sử dụng nó để tạo ra các dự đoán trên cả tập huấn luyện và tập kiểm tra:

```
y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
```

- **lr:** Đây là đối tượng đại diện cho mô hình hồi quy tuyến tính đã được huấn luyện trước đó.
- **.predict():** Đây là một phương thức của đối tượng mô hình, được sử dụng để tạo ra dự đoán cho các điểm dữ liệu mới.
- **X_train:** Tập huấn luyện, chứa các đặc trưng đầu vào được sử dụng để huấn luyện mô hình.
- **X_test:** Tập kiểm tra, chứa các đặc trưng đầu vào được sử dụng để đánh giá hiệu suất của mô hình.
- **y_pred_train:** Biến này sẽ lưu trữ các dự đoán của mô hình cho tập huấn luyện.
- **y_pred_test:** Biến này sẽ lưu trữ các dự đoán của mô hình cho tập kiểm tra.

Hãy nhớ rằng mô hình đã được huấn luyện dựa trên phiên bản biến đổi của biến mục tiêu. Do đó, chúng ta cần thực hiện phép biến đổi Yeo-Johnson nghịch đảo để chuyển đổi các dự đoán của mô hình về định dạng ban đầu của biến mục tiêu:

```
y_pred_train = pt.inverse_transform(y_pred_train.reshape(-1, 1))[:, 0]
projectpro.checkpoint('1e808c')
y_pred_test = pt.inverse_transform(y_pred_test.reshape(-1, 1))[:, 0]
```

- **pt:** Đối tượng PowerTransformer đã được sử dụng trước đó để biến đổi biến mục tiêu.
- **.inverse_transform():** Phương thức này thực hiện phép biến đổi nghịch đảo, đưa dữ liệu về thang đo ban đầu.
- **y_pred_train.reshape(-1, 1):** Thay đổi hình dạng của **y_pred_train** thành một mảng 2 chiều với một cột. Điều này là cần thiết để **inverse_transform** hoạt động chính xác.
- **[:, 0]:** Lấy cột đầu tiên (và duy nhất) từ kết quả của **inverse_transform**, đảm bảo **y_pred_train** là một mảng 1 chiều.
- **projectpro.checkpoint('1e808c'):** Có thể là một hàm để lưu tiến trình hoặc tạo điểm kiểm tra, không liên quan trực tiếp đến việc biến đổi dữ liệu.

- Các bước tương tự được lặp lại cho **y_pred_test**.

Với các dự đoán của chúng ta, chúng ta có thể đánh giá mô hình của mình:

```
base_perf_train = calc_model_performance(y_train, y_pred_train)
```

- **base_perf_train:** Biến này sẽ lưu trữ các chỉ số hiệu suất của mô hình cơ sở trên tập huấn luyện.
- **calc_model_performance:** Đây là một hàm dùng để tính toán các chỉ số hiệu suất khác nhau cho một mô hình nhất định. Hàm này có thể nhận giá trị mục tiêu thực tế và giá trị mục tiêu dự đoán làm đầu vào.
- **y_train:** Giá trị mục tiêu thực tế cho tập huấn luyện.
- **y_pred_train:** Giá trị mục tiêu dự đoán cho tập huấn luyện, được tạo ra bởi mô hình cơ sở.

base_perf_test là một biến được sử dụng để lưu trữ các chỉ số hiệu suất của mô hình cơ sở (baseline model) trên tập kiểm tra (test set).

```
base_perf_train
```

```
{'Root Mean Squared Error': 5964.030079525293,
 'Mean Squared Error': 35569654.78948248,
 'Mean Absolute Error': 4583.192074027691,
 'Mean Absolute Percentage Error': 0.7487184929524602,
 'R Squared': 0.7572131565075644}
```

```
base_perf_test = calc_model_performance(y_test, y_pred_test)
```

- **base_perf_test:** Biến này sẽ lưu trữ các chỉ số hiệu suất của mô hình cơ sở trên tập kiểm tra. Các chỉ số này có thể bao gồm R-bình phương, RMSE, MAE, v.v. tùy thuộc vào hàm calc_model_performance.
- **calc_model_performance:** Đây là một hàm (có thể được định nghĩa ở một nơi khác trong mã của bạn) được sử dụng để tính toán các chỉ số hiệu suất cho một mô hình. Hàm này nhận giá trị mục tiêu thực tế và giá trị mục tiêu dự đoán làm đầu vào.
- **y_test:** Giá trị mục tiêu thực tế cho tập kiểm tra. Đây là dữ liệu mà mô hình chưa từng thấy trước đó.
- **y_pred_test:** Giá trị mục tiêu dự đoán cho tập kiểm tra, được tạo ra bởi mô hình cơ sở.

base_perf_test là một biến được sử dụng để lưu trữ các chỉ số hiệu suất của mô hình cơ sở (baseline model) trên tập kiểm tra (test set).

```
base_perf_test
```

```
{'Root Mean Squared Error': 5752.477398059849,
 'Mean Squared Error': 33090996.215189416,
 'Mean Absolute Error': 4534.422635060568,
 'Mean Absolute Percentage Error': 0.756537151185135,
 'R Squared': 0.774191723271553}
```

Vậy chỉ số đánh giá cho mô hình của chúng ta (RMSE) là khoảng 5964 trên tập huấn luyện và khoảng 5752 trên tập kiểm tra.

5.3.10 Kiểm tra tính chuẩn của phần dư

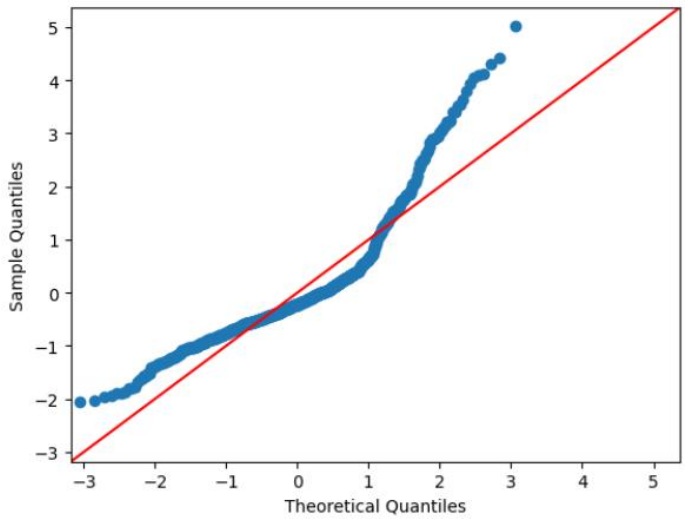
Chúng ta có thể kiểm tra tính chuẩn của phần dư bằng cách sử dụng biểu đồ QQ (quantile-quantile). Biểu đồ này vẽ giá trị của mỗi phân vị thực tế (từ dữ liệu) so với phân vị lý thuyết (giả sử phân phối chuẩn). Nếu dữ liệu phân phối chuẩn hoàn hảo, bạn sẽ thấy các điểm dữ liệu nằm trên đường thẳng.
Chúng ta cũng sẽ sử dụng biểu đồ histogram để trực quan hóa phần dư một cách dễ hiểu hơn.

```
residuals_train = y_train - y_pred_train
residuals_test = y_test - y_pred_test
```

- **residuals_train:** Biến này sẽ lưu trữ phần dư của mô hình trên tập huấn luyện.
- **residuals_test:** Biến này sẽ lưu trữ phần dư của mô hình trên tập kiểm tra.
- **y_train:** Giá trị mục tiêu thực tế cho tập huấn luyện.
- **y_pred_train:** Giá trị mục tiêu dự đoán của mô hình trên tập huấn luyện.
- **y_test:** Giá trị mục tiêu thực tế cho tập kiểm tra.
- **y_pred_test:** Giá trị mục tiêu dự đoán của mô hình trên tập kiểm tra.

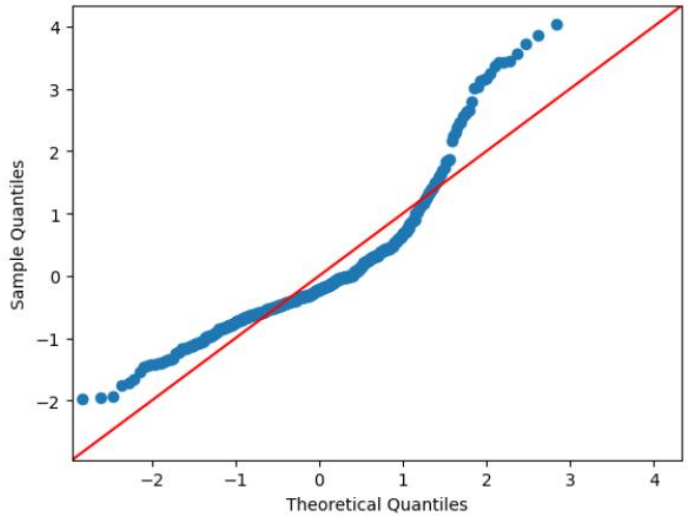
```
fig = sm.qqplot(
    residuals_train,
    fit=True,
    line='45'
)
```

- **sm.qqplot:** Hàm này, có lẽ từ thư viện statsmodels, được sử dụng để vẽ biểu đồ Q-Q.
- **residuals_train:** Đây là dữ liệu được sử dụng để vẽ biểu đồ, trong trường hợp này là phần dư trên tập huấn luyện.
- **fit=True:** Tham số này cho biết hàm sẽ tự động điều chỉnh tỷ lệ của trục để các điểm dữ liệu nằm gọn trong biểu đồ.
- **line='45':** Tham số này chỉ định rằng một đường tham chiếu với góc 45 độ sẽ được vẽ trên biểu đồ. Đường này đại diện cho phân phối chuẩn lý thuyết.



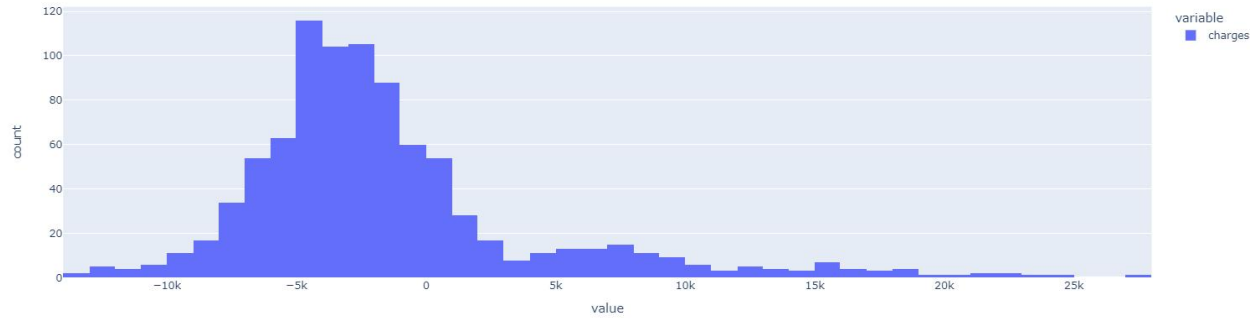
```
fig = sm.qqplot(
    residuals_test,
    fit=True,
    line='45'
)
```

- **sm.qqplot:** Hàm này, có thể từ thư viện statsmodels, được sử dụng để vẽ biểu đồ Q-Q.
- **residuals_test:** Đây là dữ liệu được sử dụng để vẽ biểu đồ, trong trường hợp này là phần dư trên tập kiểm tra.
- **fit=True:** Tham số này cho biết hàm sẽ tự động điều chỉnh tỷ lệ của trục để các điểm dữ liệu nằm gọn trong biểu đồ.
- **line='45':** Tham số này chỉ định rằng một đường tham chiếu với góc 45 độ sẽ được vẽ trên biểu đồ. Đường này đại diện cho phân phối chuẩn lý thuyết.




```
plot_residuals(y_true=y_train, y_pred=y_pred_train)
```

- **plot_residuals**: Đây là một hàm được định nghĩa ở một nơi khác trong mã của bạn, được thiết kế để vẽ biểu đồ phần dư.
- **y_true=y_train**: Tham số này cung cấp cho hàm giá trị thực tế của biến mục tiêu trên tập huấn luyện.
- **y_pred=y_pred_train**: Tham số này cung cấp cho hàm giá trị dự đoán của mô hình trên tập huấn luyện.

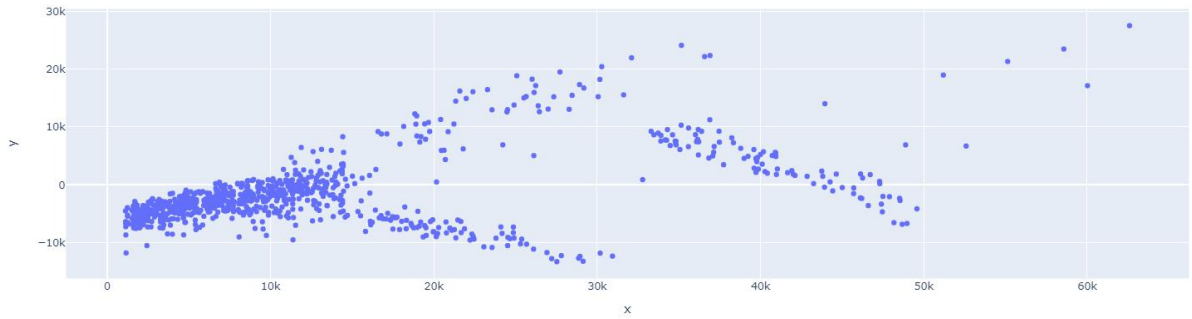


5.3.11 Kiểm tra tính đồng nhất phương sai

Chúng ta có thể kiểm tra tính đồng nhất phương sai bằng cách sử dụng biểu đồ phân tán, trong đó biến mục tiêu được hiển thị dọc theo trục x và phần dư được hiển thị dọc theo trục y. Chúng ta mong đợi các điểm dữ liệu được phân bố đồng đều trên trục y khi x (tức là giá trị mục tiêu) tăng lên:

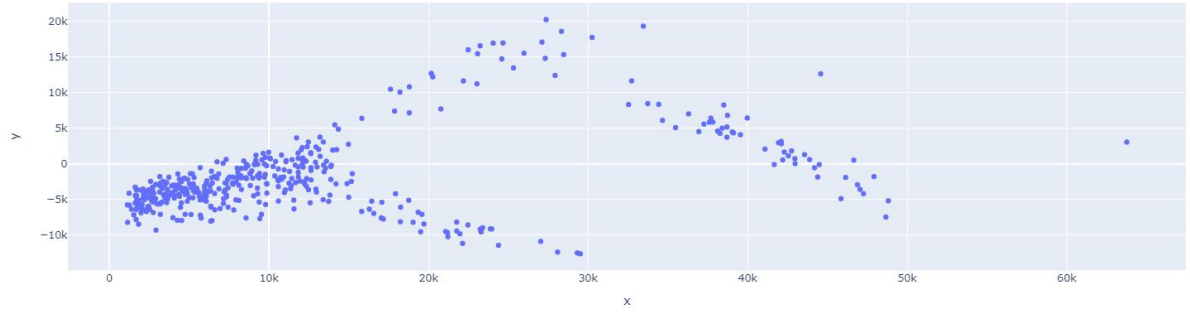
```
px.scatter(x=y_train, y=residuals_train)
```

- **px.scatter:** Hàm này từ thư viện plotly.express được sử dụng để tạo biểu đồ phân tán.
- **x=y_train:** Tham số này chỉ định rằng biến mục tiêu (y_train) sẽ được vẽ trên trục x (trục ngang).
- **y=residuals_train:** Tham số này chỉ định rằng phần dư (residuals_train) sẽ được vẽ trên trục y (trục dọc).




```
px.scatter(x=y_test, y=residuals_test)
```

- px.scatter: Đây là một hàm từ thư viện plotly.express được sử dụng để tạo biểu đồ phân tán.
- x=y_train: Tham số này chỉ định rằng biến mục tiêu (y_train) sẽ được vẽ trên trục hoành (trục x).
- y=residuals_train: Tham số này chỉ định rằng phần dư (residuals_train) sẽ được vẽ trên trục tung (trục y).



Mô hình của chúng ta thể hiện tính không đồng nhất phương sai đáng kể (tức là phương sai của phần dư không đồng nhất đối với biến mục tiêu). Vì đây là mô hình cơ sở, chúng ta sẽ không dành thêm thời gian để cố gắng cải thiện điều này.

5.4 Cải thiện mô hình tuyến tính cơ sở

Bây giờ, hãy thử cải thiện mô hình hồi quy tuyến tính của chúng ta bằng cách huấn luyện một mô hình phi tuyến tính. Ở đây, phi tuyến tính đề cập đến thực tế là mô hình có thể học các mối quan hệ phi tuyến tính từ dữ liệu.

Trước khi chuyển sang XGBoost Regressor, hãy cùng tìm hiểu về Cây quyết định.

Cây quyết định

Cây quyết định, thường được gọi là CART hoặc Cây phân loại và hồi quy, là các thuật toán học máy có giám sát linh hoạt có khả năng thực hiện cả bài toán hồi quy và phân loại. và cũng có thể được sử dụng cho các hoạt động với nhiều lớp. Cái tên ngụ ý rằng nó sử dụng một sơ đồ luồng giống như cây để hiển thị các dự đoán đến từ một chuỗi các phân tách dựa trên đặc trưng. Nó bắt đầu với một nút gốc và kết thúc bằng một quyết định lá. Nó có thể được sử dụng để miêu tả các quyết định và quá trình ra quyết định một cách trực quan và rõ ràng. Gốc của một cây quyết định nằm ở trên cùng và gốc này được chia thành nhiều nút.

Nói một cách dễ hiểu, cây quyết định không gì khác hơn là một chuỗi các câu lệnh if-else. Nó xác định xem điều kiện có đúng hay không, và nếu đúng, nó sẽ tiến hành đến nút tiếp theo liên quan đến lựa chọn đó.

Cây quyết định và Quá khớp

Việc chọn các đặc trưng để sử dụng và các điều kiện để sử dụng cho việc phân tách, cũng như hiểu biết về thời điểm dừng, đều là một phần của quá trình phát triển cây. Tất cả các đặc trưng đều được xem xét, và nhiều điểm phân tách được kiểm tra bằng cách sử dụng hàm chi phí và nhiều nhánh cũng có thể phản ánh nhiễu hoặc ngoại lệ trong dữ liệu huấn luyện. Phân tách có entropy thấp nhất được chọn. Đây là một ví dụ về thuật toán đệ quy vì các nhóm được tạo ra có thể được chia nhỏ bằng cách sử dụng cùng một phương pháp và nó còn được gọi là thuật toán tham lam vì nó sẽ vét cạn tất cả các đặc trưng và điều kiện để đạt được chi phí thấp nhất, ngay cả khi thuật toán cuối cùng bị quá khớp với tập dữ liệu. Trong cây quyết định, quá khớp xảy ra khi cây được thiết kế để phù hợp hoàn hảo với tất cả các mẫu trong tập dữ liệu huấn luyện.

Hàm chi phí hoặc trong trường hợp này là Độ lợi thông tin tìm ra đặc trưng để phân tách sao cho tạo ra các nhánh đồng nhất nhất có thể. Kết quả là, bạn có thể ánh xạ lựa chọn của một lớp đồng nhất cụ thể ở cuối với tất cả các đặc trưng liên quan từ trên xuống, và cây sẽ liên kết lớp đồng nhất đó với tập hợp các đặc trưng đó. Vì một tập dữ liệu thường có một số đặc trưng, nên nó dẫn đến một số lượng lớn các phân tách, từ đó tạo ra một cây khổng lồ. Các cây như vậy rất phức tạp và có thể dẫn đến quá khớp vì cây sẽ học từng giá trị đặc trưng cho một quan sát cụ thể khi cây phát triển đến độ sâu tối đa và khi một quan sát mới được kiểm tra, nó sẽ tạo ra các lỗi lớn. Do đó, cây quyết định đặc biệt dễ bị quá khớp.

Bây giờ, chúng ta hãy cùng tìm hiểu về phương pháp Ensemble, đặc biệt là Bagging và Boosting. Bagging và Boosting là các loại kỹ thuật học tập tổng hợp (ensemble learning) giúp giảm phương sai của một bộ ước lượng đơn lẻ bằng cách kết hợp sức mạnh của nhiều bộ học. Sự khác biệt đáng kể là bagging liên quan đến việc lấy trung bình kết quả của nhiều bộ học, trong khi boosting bao hàm việc xây dựng tuần tự, tạo ra các bộ học mạnh bằng cách phạt lỗi đối với các bộ học yếu ban đầu. Hãy xem video tham khảo bên dưới để biết thêm chi tiết.

Boosting

Thuật ngữ 'boosting' đề cập đến một tập hợp các thuật toán giúp các bộ học yếu trở nên mạnh mẽ. Không giống như nhiều mô hình học máy khác, các thuật toán boosting nhằm mục đích cải thiện khả năng dự đoán bằng cách huấn luyện một loạt các mô hình yếu, mỗi mô hình sẽ sửa chữa các lỗi của mô hình trước đó. Điều này được thực hiện bằng cách xâu chuỗi các mô hình yếu lại với nhau để tạo ra một mô hình mạnh. Để bắt đầu, một mô hình được tạo ra bằng cách sử dụng dữ liệu huấn luyện. Sau đó, mô hình thứ hai được tạo ra, mô hình này cố gắng sửa chữa các lỗi trong mô hình trước đó. Phương pháp này được lặp lại cho đến khi toàn bộ tập dữ liệu huấn luyện được dự đoán chính xác hoặc số lượng mô hình tối đa được chỉ định đã được tạo.

Để hiểu rõ về boosting, điều quan trọng cần nhớ là boosting là một phương pháp chung, không phải là một mô hình cụ thể. Boosting liên quan đến việc chỉ định một mô hình yếu, chẳng hạn như hồi quy hoặc cây quyết định, và sau đó cải thiện nó. Trong Học tập tổng hợp (Ensemble Learning), điểm khác biệt chính giữa Bagging và Boosting là trong bagging, các bộ học yếu được huấn luyện đồng thời, nhưng trong boosting, chúng được huấn luyện tuần tự. Điều này có nghĩa là mỗi lần lặp lại mô hình mới sẽ tăng trọng số của dữ liệu được phân loại sai của mô hình trước đó. Việc phân phối lại trọng số này giúp thuật toán xác định các tham số mà nó nên tập trung vào để nâng cao hiệu suất.
Cả hai kỹ thuật Ensemble cũng được sử dụng theo những cách khác nhau. Ví dụ, các phương pháp Bagging thường được sử dụng trên các bộ học yếu có phương sai lớn và độ lệch thấp như cây quyết định vì chúng có xu hướng quá khớp, trong khi các phương pháp Boosting được sử dụng khi có phương sai thấp và độ lệch cao. Mặc dù Bagging có thể giúp ngăn ngừa quá khớp, nhưng các phương pháp Boosting dễ bị quá khớp hơn do một thực tế đơn giản là chúng tiếp tục xây dựng trên các bộ học yếu và tiếp tục giảm thiểu lỗi. Điều này có thể dẫn đến quá khớp trên dữ liệu huấn luyện, nhưng việc chỉ định một số lượng mô hình hợp lý để tạo ra hoặc điều chỉnh siêu tham số, điều chuẩn (regularization) có thể giúp ích trong trường hợp này, nếu gặp phải quá khớp.
Gradient Boosting
Ý tưởng chính đằng sau kỹ thuật này là phát triển các mô hình một cách tuần tự, với mỗi mô hình cố gắng giảm thiểu các lỗi của mô hình trước đó. Mô hình cộng, hàm mất mát và bộ học yếu là ba thành phần cơ bản của Gradient Boosting.
Phương pháp này cung cấp một cách giải thích trực tiếp về boosting dưới dạng tối ưu hóa số học của hàm mất mát bằng cách sử dụng Gradient Descent. Chúng ta sử dụng Gradient Boosting Regressor khi cột mục tiêu là liên tục và Gradient Boosting Classifier khi nhiệm vụ là bài toán phân loại. "Hàm mất mát" là điểm khác biệt duy nhất giữa hai phương pháp này. Mục tiêu là sử dụng gradient descent để giảm thiểu hàm mất mát này bằng cách thêm các bộ học yếu. Vì nó dựa trên các hàm mất mát, đối với các bài toán hồi quy, sẽ sử dụng Sai số bình phương trung bình (MSE), và đối với các bài toán phân loại, sẽ sử dụng log-likelihood.

5.4.1 Giới thiệu về mô hình phi tuyến tính-XGBoost

XGBoost (eXtreme Gradient Boost) là một thuật toán dựa trên cây quyết định và được tăng cường gradient. Nó là thuật toán có hiệu suất hàng đầu trong nhiều cuộc thi Kaggle, vì vậy nó là một ứng cử viên lý tưởng để áp dụng cho trường hợp sử dụng này.
XGBoost cho hồi quy được huấn luyện bằng quy trình sau:

1. Đưa ra dự đoán ban đầu (thường chỉ là giá trị **0.5**)
2. Tính toán phần dư của mỗi quan sát (so với dự đoán ban đầu này)
3. Khớp một cây quyết định hồi quy với các phần dư:

    1. Tính điểm tương đồng cho các quan sát trong mỗi nút lá:

- $S = \frac{\sum(R)^2}{N_R+\lambda}$, trong đó R là giá trị phần dư, N_R là số lượng phần dư và λ là tham số điều chuẩn.

    2. Sau đó thử phân tách các nút lá hơn nữa để cải thiện việc phân cụm các phần dư tương tự:

- Đối với mỗi lần phân tách, điểm tương đồng được tính cho các nút lá trái và phải.
- Sự cải thiện bằng cách phân tách được định lượng bằng cách sử dụng độ lợi:

$$G = S_{left} + S_{right} - S_{root}$$

- Chọn phân tách tối đa hóa độ lợi.

3. Lặp lại ba bước này cho đến khi đạt được số lượng quan sát tối thiểu trên mỗi nút lá (xem tham số **min_child_weight** trong XGBoost API) hoặc cho đến khi đạt được độ sâu tối đa (xem tham số **max_depth** trong XGBoost API).
4. Sau đó, cây quyết định hồi quy được cắt tỉa dựa trên giá trị độ lợi của mỗi lần phân tách, sử dụng ngưỡng γ (xem tham số **gamma** trong XGBoost API):

- Đối với lần phân tách thấp nhất trong cây, nếu G < γ, thì lần phân tách đó sẽ bị loại bỏ; nếu nó dương, lần phân tách sẽ được giữ lại.
- Lặp lại quá trình này cho mỗi lần phân tách, di chuyển lên phía gốc (và bao gồm cả chính gốc)

5. Sau khi cây quyết định hồi quy được cắt tỉa, hãy tính toán giá trị đầu ra của mỗi nút lá:

$$O = \frac{\sum R}{N_R + \lambda}$$

- Lưu ý: khi **λ>0**, nó sẽ giảm lượng mà một nút lá riêng lẻ thêm vào dự đoán tổng thể, làm giảm độ nhạy của dự đoán cuối cùng đối với dự đoán của cây riêng lẻ (xem tham số **reg_lambda** trong XGBoost API).

6. Sau đó, kết hợp dự đoán ban đầu với dự đoán từ cây quyết định hồi quy (được điều chỉnh tỷ lệ bằng tốc độ học, η - xem tham số **learning_rate** trong XGBoost API) để có được dự đoán mới:

$$Pred_{new} = Pred_{init} + \eta * Pred_{residual}$$

7. Lặp lại các bước 1-6 cho đến khi phần dư đạt đến một ngưỡng nhất định hoặc cho đến khi đạt được số lượng cây tối đa (xem tham số **n_estimators** trong XGBoost API).

5.4.2 Tiền xử lý

Hãy tạo một tập huấn luyện và tập kiểm tra mới (vì tập huấn luyện và tập kiểm tra mà chúng ta đã tạo cho mô hình tuyến tính cơ sở đã được biến đổi).
Lưu ý: chúng ta sử dụng cùng một giá trị cho tham số **random_seed** để đảm bảo các quan sát trong các tập huấn luyện và tập kiểm tra này giống với các quan sát trong tập huấn luyện và tập kiểm tra được sử dụng cho mô hình tuyến tính cơ sở:

Chia thành tập huấn luyện và kiểm tra

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.33,
    random_state=42
)
```

1. **train_test_split:** Hàm này thực hiện việc chia tách dữ liệu.
2. **test_size=0.33:** Tham số này chỉ định rằng 33% dữ liệu của bạn sẽ được phân bổ cho tập kiểm tra và 67% còn lại sẽ được sử dụng để huấn luyện. Bạn có thể điều chỉnh tỷ lệ này khi cần thiết.

3. **random_state=42:** Tham số này đảm bảo rằng việc chia tách có thể tái tạo được. Nếu bạn chạy lại mã với cùng random_state, bạn sẽ nhận được cùng một kết quả chia tách mỗi lần. Điều này quan trọng để có kết quả nhất quán.
4. **X_train, X_test, y_train, y_test:** Đây là các biến đầu ra lưu trữ các tập huấn luyện và kiểm tra kết quả:

- X_train: Các đặc trưng đầu vào để huấn luyện mô hình.
- X_test: Các đặc trưng đầu vào để kiểm tra mô hình.
- y_train: Biến mục tiêu để huấn luyện mô hình.
- y_test: Biến mục tiêu để kiểm tra mô hình.

Mã hóa

```
ohe = OneHotEncoder(use_cat_names=True)
X_train = ohe.fit_transform(X_train)
projectpro.checkpoint('1e808c')
X_test = ohe.transform(X_test)
```

1. **ohe = OneHotEncoder(use_cat_names=True):**

- Khởi tạo một đối tượng OneHotEncoder với **use_cat_names=True**.
- OneHotEncoder là một lớp từ thư viện scikit-learn được sử dụng để thực hiện mã hóa one-hot.
- **use_cat_names=True** chỉ định rằng tên của các danh mục (**categories**) sẽ được sử dụng làm tên cột trong dữ liệu được mã hóa, giúp dữ liệu dễ hiểu hơn.

2. **X_train = ohe.fit_transform(X_train):**

- fit_transform được áp dụng trên tập huấn luyện (**X_train**).
- fit: "Học" các danh mục duy nhất (**unique categories**) từ các đặc trưng phân loại trong **X_train**.
- **transform:** Biến đổi **X_train** bằng cách áp dụng mã hóa one-hot, tạo ra các cột mới đại diện cho mỗi danh mục. Các giá trị trong các cột này sẽ là 0 hoặc 1, cho biết sự hiện diện hoặc vắng mặt của danh mục tương ứng trong mỗi mẫu dữ liệu.

3. **projectpro.checkpoint('1e808c'):**

- Đây có vẻ là một hàm hoặc phương thức cụ thể của **projectpro** (có thể là một nền tảng hoặc thư viện nào đó) được sử dụng để lưu trữ điểm kiểm tra (checkpoint).
- Điểm kiểm tra cho phép bạn lưu trạng thái của mô hình hoặc quá trình huấn luyện, để bạn có thể tiếp tục từ điểm đó sau này nếu cần.

4. **X_test = ohe.transform(X_test):**

- Áp dụng mã hóa one-hot trên tập kiểm tra (**X_test**) bằng cách sử dụng đối tượng ohe đã được "huấn luyện" trước đó trên **X_train**.
- Điều này đảm bảo rằng mã hóa được áp dụng một cách nhất quán cho cả tập huấn luyện và tập kiểm tra, sử dụng cùng một tập hợp các danh mục đã học từ **X_train**.

5.4.3 Sử dụng pipeline của Sklearn để tối ưu hóa quy trình huấn luyện mô hình

Như đã đề cập trong phần giới thiệu, thuật toán XGBoost có một số tham số có thể ảnh hưởng đến hiệu suất dự đoán của mô hình. Thay vì cố gắng tối ưu hóa các tham số này bằng tay, chúng ta có thể tận dụng lớp Pipeline của Sklearn, cùng với lớp BayesSearchCV của Skopt, để tối ưu hóa chúng một cách tự động.
Chúng ta cũng sẽ sử dụng Loại bỏ Đặc trưng Đệ quy (RFE) trong quy trình này để tối ưu hóa các đặc trưng được sử dụng bởi mô hình XGBoost cuối cùng. RFE hoạt động bằng cách khớp một mô hình với tất cả các đặc trưng, sau đó tính toán mức độ quan trọng của đặc trưng. Đặc trưng ít quan trọng nhất sẽ bị loại bỏ, sau đó một mô hình khác được huấn luyện bằng cách sử dụng tập dữ liệu này. Quá trình này được lặp lại cho đến khi đạt được số lượng đặc trưng mong muốn để lựa chọn.
Đầu tiên, hãy tạo các instance của RFE và XGBoost regressor. Chúng ta cũng sẽ sử dụng một XGBoost regressor làm mô hình trong RFE của chúng ta.

```
rfe = RFE(estimator=XGBRegressor())
xgb = XGBRegressor()
```

1. **rfe = RFE(estimator=XGBRegressor()):**
   - Khởi tạo một đối tượng **RFE** (Recursive Feature Elimination).
   - **estimator=XGBRegressor():** Chỉ định **XGBRegressor** là mô hình sẽ được sử dụng bên trong RFE để đánh giá tầm quan trọng của các đặc trưng.

2. **xgb = XGBRegressor():**
   - Khởi tạo một đối tượng **XGBRegressor**.
   - Đây là mô hình dự đoán chính sẽ được sử dụng sau khi RFE đã lựa chọn các đặc trưng quan trọng nhất.

Bây giờ chúng ta tạo pipeline bằng cách chỉ định danh sách các quy trình tuần tự mà chúng ta muốn chạy. Đầu ra của mỗi bước được chuyển đến bước tiếp theo, với bước cuối cùng là một estimator (tức là mô hình). Trong trường hợp này, chúng ta muốn:

- Áp dụng lựa chọn đặc trưng (thông qua RFE)
- Huấn luyện XGBoost regressor

Chúng ta thực hiện việc này bằng cách tạo một danh sách các tuple, trong đó phần tử đầu tiên của mỗi tuple là nhãn cho bước đó và phần tử thứ hai là lớp để chạy:

```
steps = [
    ('rfe', rfe),
    ('xgb', xgb)
]
```

steps: Một biến lưu trữ danh sách các bước trong pipeline.
Mỗi bước trong pipeline được biểu diễn bằng một tuple: **('tên bước', đối tượng bước).**

- **('rfe', rfe):** Bước đầu tiên trong pipeline có tên là 'rfe' và sử dụng đối tượng rfe (đã được định nghĩa trước đó, thường là một đối tượng RFE - Recursive Feature Elimination) để thực hiện lựa chọn đặc trưng.
- **('xgb', xgb):** Bước thứ hai trong pipeline có tên là 'xgb' và sử dụng đối tượng xgb (đã được định nghĩa trước đó, thường là một đối tượng XGBRegressor) để huấn luyện mô hình XGBoost.

Sau đó, chúng ta truyền danh sách này vào lớp Pipeline:

```
pipe = Pipeline(steps)
```

Khi phương thức fit của pipeline được gọi, nó sẽ truyền các đặc trưng và mục tiêu đến phương thức fit_transform của RFE. Đầu ra của phương thức này sẽ được truyền đến phương thức fit của XGBoost regressor.
Bây giờ chúng ta đã định nghĩa pipeline, chúng ta có thể thiết lập lớp BayesSearchCV để tối ưu hóa các tham số của pipeline. Đầu tiên, chúng ta cần định nghĩa các tham số mà chúng ta muốn tối ưu hóa và không gian mà BayesSearchCV nên tìm kiếm. Lưu ý rằng vì chúng ta đang sử dụng pipeline, chúng ta cần định nghĩa các tham số liên quan đến mỗi bước bằng cách thêm tiền tố, là nhãn của bước và hai dấu gạch dưới.

```
num_features = X_train.shape[1]
search_spaces = {
    'rfe__n_features_to_select': Integer(1, num_features),
    'xgb__n_estimators': Integer(1, 500),
    'xgb__max_depth': Integer(2, 8),
    'xgb__reg_lambda': Integer(1, 200),
    'xgb__learning_rate': Real(0, 1),
    'xgb__gamma': Real(0, 2000)
}
```

1. **num_features = X_train.shape[1]:**
   - Lấy số lượng đặc trưng (features) từ tập huấn luyện (X_train).
   - X_train.shape[1] trả về số lượng cột trong X_train, đại diện cho số lượng đặc trưng.

2. **search_spaces = { ... }:**
   - Định nghĩa một từ điển (dictionary) chứa không gian tìm kiếm cho các siêu tham số.
   - Mỗi khóa (key) trong từ điển tương ứng với một siêu tham số cụ thể của pipeline.
   - Mỗi giá trị (value) tương ứng với một đối tượng định nghĩa phạm vi và kiểu dữ liệu của siêu tham số đó.

Sau khi không gian tìm kiếm cho mỗi tham số được xác định, chúng ta truyền nó, cùng với pipeline của chúng ta, đến lớp BayesSearchCV.

1. Lớp này hoạt động bằng cách: Tạo một tập dữ liệu cross-validation gồm n folds, trong đó mỗi fold có cùng số lượng quan sát. Ví dụ: đối với 3 folds, tập dữ liệu gốc được chia ngẫu nhiên thành 3 tập con.
2. Các bước sau được áp dụng trên tất cả các folds:
   - Tạo một tập huấn luyện con bao gồm n-1 folds và một tập validation con bao gồm fold còn lại.
   - Huấn luyện pipeline bằng cách sử dụng một tập hợp các tham số nhất định trên tập huấn luyện con.
   - Sử dụng pipeline đã được huấn luyện này để đưa ra dự đoán trên tập validation con và tính toán độ đo đánh giá (trong trường hợp này là RMSE).
3. Tính toán giá trị trung bình của độ đo đánh giá trên tất cả các tập validation con.
4. Thuật toán sau đó sử dụng phương pháp Bayesian để chọn tập hợp tham số tiếp theo sẽ tối đa hóa giá trị trung bình của độ đo đánh giá trên tất cả các tập validation con.
5. Các bước 2-4 được lặp lại cho đến khi đạt đến một ngưỡng nhất định (xem tham số n_iter trong lớp BayesSearchCV).

5.4.4 BayesSearchCV

Phương pháp này sử dụng Tối ưu hóa Bayesian từng bước để khảo sát các siêu tham số hứa hẹn nhất trong không gian vấn đề. Trong không gian vấn đề lớn, Tối ưu hóa Bayesian xác định giá trị cực tiểu cho một hàm mục tiêu. Bên trong, nó sử dụng hồi quy quá trình Gaussian trên hàm mục tiêu để thực hiện việc này. Trong ví dụ của chúng ta, hàm mục tiêu là tìm đầu ra mô hình tối ưu dựa trên các tham số mô hình mà chúng ta chỉ định. Phương pháp Tối ưu hóa Bayes có lợi thế là cung cấp phạm vi giá trị có thể lớn hơn đáng kể vì nó tự động khám phá các vùng triển vọng nhất và loại bỏ các vùng kém triển vọng hơn theo thời gian.
Việc tìm kiếm theo lưới đơn giản sẽ mất rất nhiều thời gian để xem xét tất cả các giá trị có thể có. Vì chúng ta di chuyển hiệu quả hơn nhiều nên chúng ta có thể tạo ra một sân chơi lớn hơn nhiều.
Tối ưu hóa Bayes theo dõi các kết quả đánh giá trước đó, được sử dụng để xây dựng một mô hình xác suất ánh xạ các siêu tham số với xác suất của một điểm số trên hàm mục tiêu. Về cơ bản, nó tìm ra các siêu tham số triển vọng nhất bằng cách xây dựng một mô hình xác suất của hàm mục tiêu.
P = (score | hyperparameters)
Hàm xác suất này hoạt động như một "hàm thay thế" cho các hàm mục tiêu. Các siêu tham số được kiểm tra trên hàm này và những siêu tham số nổi bật sau đó được kiểm tra trên hàm mục tiêu thực sự, là một hàm giảm thiểu sai số. Sau đó, kết quả sẽ cập nhật lại hàm thay thế và theo cách này, tối ưu hóa Bayes sẽ chọn các tham số một cách có căn cứ.

```
xgb_bs_cv = BayesSearchCV(
    estimator=pipe,
    search_spaces=search_spaces,
    scoring='neg_root_mean_squared_error',
    n_iter=75,
    cv=3,
    n_jobs=-1,
    verbose=1,
    random_state=0
)
projectpro.checkpoint('1e808c')
```

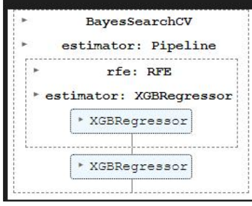
- **estimator=pipe:** Đây là nơi chỉ định pipeline mô hình (pipe) sẽ được tối ưu hóa. Pipeline có thể bao gồm các bước tiền xử lý dữ liệu và mô hình XGBoost (XGBRegressor).
- **search_spaces=search_spaces:** Đây là nơi cung cấp phạm vi giá trị siêu tham số để khám phá trong quá trình tối ưu hóa, được định nghĩa trong biến search_spaces. Biến này chỉ định không gian tìm kiếm cho các siêu tham số, bao gồm kiểu dữ liệu (ví dụ: Real, Categorical, Integer) và phạm vi.
- **scoring='neg_root_mean_squared_error':** Đây là nơi đặt chỉ số đánh giá để tối ưu hóa thành Sai số bình phương trung bình âm (RMSE). BayesSearchCV nhằm mục đích tối đa hóa chỉ số đánh giá, do đó dấu âm được sử dụng để giảm thiểu RMSE.
- **n_iter=75:** Đây là nơi giới hạn số lần lặp tối ưu hóa là 75. Thuật toán sẽ đánh giá 75 kết hợp siêu tham số khác nhau.
- **cv=3:** Đây là nơi thực hiện kiểm định chéo 3 fold trong quá trình tối ưu hóa. Dữ liệu được chia thành 3 fold và mô hình được huấn luyện và đánh giá 3 lần, với mỗi fold đóng vai trò là tập kiểm định một lần.

- **n_jobs=-1:** Đây là nơi sử dụng tất cả các lõi CPU có sẵn để tính toán song song, giúp tăng tốc quá trình tối ưu hóa.
- **verbose=1:** Đây là nơi hiển thị cập nhật tiến trình trong quá trình tối ưu hóa, cung cấp thông tin về quá trình tìm kiếm.
- **random_state=0:** Đây là nơi đảm bảo kết quả có thể tái tạo bằng cách cố định seed ngẫu nhiên. Sử dụng cùng một seed ngẫu nhiên sẽ dẫn đến cùng một quá trình tìm kiếm siêu tham số và huấn luyện mô hình.
- **projectpro.checkpoint('1e808c'):** Đây có thể là một hàm tùy chỉnh dành riêng cho môi trường ProjectPro, có khả năng để lưu tiến trình hoặc điểm kiểm tra trong quá trình thực thi mã.

Bây giờ chúng ta có thể bắt đầu quá trình tối ưu hóa tham số bằng cách sử dụng tập huấn luyện của mình:

```
xgb_bs_cv.fit(
    X_train,
    y_train,
)
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
...
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```



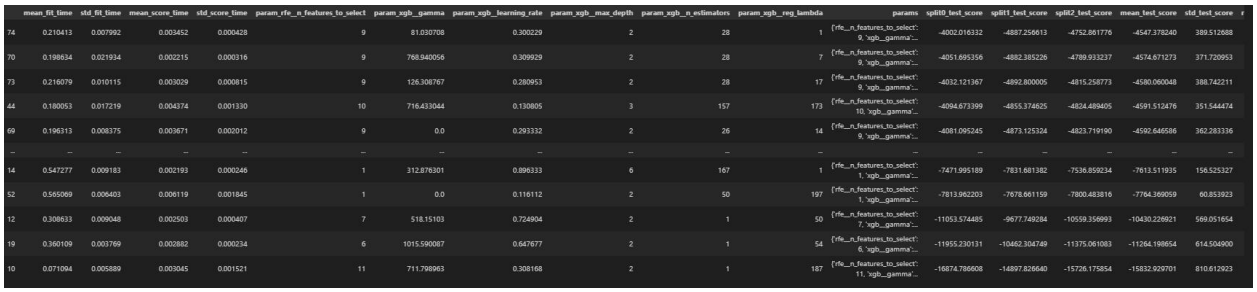
5.4.5 Đánh giá mô hình

Trước tiên, hãy xem xét cách mỗi tập hợp tham số hoạt động trên mỗi fold. Mỗi bản ghi trong tập dữ liệu tương ứng với một tập hợp tham số đã được kiểm tra. Chúng ta xếp hạng theo rank_test_score để đảm bảo tập hợp tham số hoạt động tốt nhất được hiển thị ở trên cùng:

```
cv_results = pd.DataFrame(xgb_bs_cv.cv_results_).sort_values('rank_test_score')
```

```
cv_results
```

- **xgb_bs_cv.cv_results_:** Thuộc tính này của đối tượng **BayesSearchCV** lưu trữ một từ điển chứa thông tin chi tiết về kết quả kiểm định chéo cho mỗi tổ hợp siêu tham số được đánh giá.
- **.sort_values('rank_test_score'):** Phương thức này sắp xếp **DataFrame** theo thứ tự tăng dần dựa trên các giá trị trong cột **'rank_test_score'.**



	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_rfe__n_features_to_select	param_xgb__gamma	param_xgb__learning_rate	param_xgb__max_depth	param_xgb__n_estimators	param_xgb__reg_lambda	params	split0_test_score	split1_test_score	split2_test_score	mean_test_score	std_test_score	r
74	0.210413	0.007992	0.003452	0.000428	9	81.030708	0.300229	2	28	1	{'rfe__n_features_to_select': 9, 'xgb__gamma':...	-4002.016332	-4887.256613	-4752.861776	-4547.378240	389.512688	
70	0.198634	0.021934	0.002215	0.000316	9	768.940056	0.309929	2	28	7	{'rfe__n_features_to_select': 9, 'xgb__gamma':...	-4051.695356	-4882.385226	-4789.933237	-4574.671273	371.720953	
73	0.216079	0.010115	0.003029	0.000815	9	126.308767	0.280953	2	28	17	{'rfe__n_features_to_select': 9, 'xgb__gamma':...	-4032.121367	-4892.800005	-4815.258773	-4580.060048	388.742211	
44	0.180053	0.017219	0.004374	0.001330	10	716.433044	0.130805	3	157	173	{'rfe__n_features_to_select': 10, 'xgb__gamma':...	-4094.673399	-4855.374625	-4824.489405	-4591.512476	351.544474	
69	0.196313	0.008375	0.003671	0.002012	9	0.0	0.293332	2	26	14	{'rfe__n_features_to_select': 9, 'xgb__gamma':...	-4081.095245	-4873.125324	-4823.719190	-4592.646586	362.283336	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
14	0.547277	0.009183	0.002193	0.000246	1	312.876301	0.896333	6	167	1	{'rfe__n_features_to_select': 1, 'xgb__gamma':...	-7471.995189	-7831.681382	-7536.859234	-7613.511935	156.525327	
52	0.565069	0.006403	0.006119	0.001845	1	0.0	0.116112	2	50	197	{'rfe__n_features_to_select': 1, 'xgb__gamma':...	-7813.962203	-7678.661159	-7800.483816	-7764.369059	60.853923	
12	0.308633	0.009048	0.002503	0.000407	7	518.15103	0.724904	2	1	50	{'rfe__n_features_to_select': 7, 'xgb__gamma':...	-11053.574485	-9677.749284	-10559.358993	-10430.226921	569.051654	
19	0.360109	0.003769	0.002882	0.000234	6	1015.590087	0.647677	2	1	54	{'rfe__n_features_to_select': 6, 'xgb__gamma':...	-11955.230131	-10462.304749	-11375.061083	-11264.198654	614.504900	
10	0.071094	0.005889	0.003045	0.001521	11	711.798963	0.308168	2	1	187	{'rfe__n_features_to_select': 11, 'xgb__gamma':...	-16874.786608	-14897.826640	-15726.175854	-15832.929701	810.612923	

Bây giờ, hãy tạo dự đoán trên cả tập huấn luyện và tập kiểm tra của chúng ta bằng cách sử dụng mô hình được huấn luyện với các tham số hoạt động tốt nhất của chúng ta:

```
y_pred_train_xgb = xgb_bs_cv.predict(X_train)
y_pred_test_xgb = xgb_bs_cv.predict(X_test)
```

Với các dự đoán của mình, chúng ta có thể đánh giá mô hình của mình:

```
xgb_perf_train = calc_model_performance(y_train, y_pred_train_xgb)

xgb_perf_train
```

- **calc_model_performance:** Đây là một hàm tùy chỉnh (có thể được định nghĩa trong tệp model_performance.py được đề cập trước đó) được sử dụng để đánh giá hiệu suất của mô hình hồi quy. Nó có thể tính toán các chỉ số khác nhau như RMSE, R-squared, MAE, v.v.
- **y_train:** Đây là biến chứa các giá trị thực tế của biến mục tiêu (charges) cho tập huấn luyện.
- **y_pred_train_xgb:** Đây là biến chứa các giá trị dự đoán của biến mục tiêu cho tập huấn luyện được tạo bởi mô hình XGBoost.

```
{'Root Mean Squared Error': 4186.661638235022,
 'Mean Squared Error': 17528135.673068758,
 'Mean Absolute Error': 2326.2427185848564,
 'Mean Absolute Percentage Error': 0.2666692801056923,
 'R Squared': 0.8803586720883821}

xgb_perf_test = calc_model_performance(y_test, y_pred_test_xgb)

xgb_perf_test
```

- **calc_model_performance:** Đây là một hàm tùy chỉnh (có thể được định nghĩa trong module model_performance đã được import trước đó) được thiết kế để tính toán các chỉ số đánh giá hiệu suất khác nhau cho một mô hình hồi quy. Hàm này nhận giá trị thực tế và giá trị dự đoán làm đầu vào.

- **y_test:** Biến này đại diện cho các giá trị thực tế của biến mục tiêu cho tập dữ liệu kiểm tra. Đây là những giá trị thực tế mà mô hình đang cố gắng dự đoán trên dữ liệu mà nó chưa từng được huấn luyện.
- **y_pred_test_xgb:** Biến này đại diện cho các dự đoán được tạo ra bởi mô hình XGBoost đã được huấn luyện trên tập dữ liệu kiểm tra.
- **xgb_perf_test:** Biến này sẽ lưu trữ kết quả đầu ra của hàm calc_model_performance, có thể là một từ điển hoặc một cấu trúc dữ liệu tương tự chứa các chỉ số đánh giá hiệu suất đã được tính toán (RMSE, R-squared, MAE, v.v.) trên tập dữ liệu kiểm tra.

```
{'Root Mean Squared Error': 4466.904446257581,
 'Mean Squared Error': 19953235.331995748,
 'Mean Absolute Error': 2483.459321456837,
 'Mean Absolute Percentage Error': 0.28813330823252775,
 'R Squared': 0.8638419449153072}
```

5.4.6 So sánh với mô hình cơ sở

Trước tiên, hãy so sánh các chỉ số đánh giá mà chúng ta đã tính toán cho mỗi mô hình:

```
perf_comp_train = compare_model_performance(base_perf_train, xgb_perf_train)
perf_comp_test = compare_model_performance(base_perf_test, xgb_perf_test)
```

```
perf_comp_train
```

	base	new	abs_improvement	perc_improvement
Root Mean Squared Error	5964.03	4186.66	-1777.37	-29.80
Mean Squared Error	35569654.79	17528135.67	-18041519.12	-50.72
Mean Absolute Error	4583.19	2326.24	-2256.95	-49.24
Mean Absolute Percentage Error	0.75	0.27	-0.48	-64.00
R Squared	0.76	0.88	0.12	15.79

```
perf_comp_test
```

	base	new	abs_improvement	perc_improvement
Root Mean Squared Error	5752.48	4466.90	-1285.58	-22.35
Mean Squared Error	33090996.22	19953235.33	-13137760.89	-39.70
Mean Absolute Error	4534.42	2483.46	-2050.96	-45.23
Mean Absolute Percentage Error	0.76	0.29	-0.47	-61.84
R Squared	0.77	0.86	0.09	11.69

- **compare_model_performance:** Đây là một hàm tùy chỉnh (có thể đã được định nghĩa trước đó trong mã của bạn) nhận các chỉ số đánh giá hiệu suất của hai mô hình làm đầu vào và tính toán sự khác biệt hoặc tỷ lệ giữa chúng để tạo điều kiện thuận lợi cho việc so sánh.
- **base_perf_train và base_perf_test:** Các biến này có thể lưu trữ các chỉ số đánh giá hiệu suất (RMSE, R-squared, v.v.) của mô hình cơ sở trên tập dữ liệu huấn luyện và tập dữ liệu kiểm tra, tương ứng. Chúng có thể đã được tính toán

trước đó trong mã của bạn bằng cách sử dụng phương pháp tương tự như đối với mô hình XGBoost.

- **xgb_perf_train và xgb_perf_test:** Như đã thảo luận trước đó, các biến này lưu trữ các chỉ số đánh giá hiệu suất của mô hình XGBoost trên tập dữ liệu huấn luyện và tập dữ liệu kiểm tra, tương ứng.
- **perf_comp_train và perf_comp_test:** Các biến này sẽ lưu trữ kết quả đầu ra của hàm compare_model_performance cho tập dữ liệu huấn luyện và tập dữ liệu kiểm tra, tương ứng. Kết quả đầu ra có thể chứa sự khác biệt hoặc tỷ lệ giữa các chỉ số đánh giá hiệu suất của mô hình cơ sở và mô hình XGBoost.

Chúng ta có thể quan sát thấy sự giảm đáng kể các lỗi và sự tăng giá trị R2 đối với mô hình XGBoost.
Quan trọng nhất, RMSE (chỉ số đánh giá mô hình mà chúng ta đã xác định ở đầu bài tập) đã giảm ~22% trên tập kiểm tra!

5.4.7 Trình bày kết quả

Là các nhà khoa học dữ liệu, chúng ta thường cần phải truyền đạt hiệu suất của một mô hình cho các bên liên quan không chuyên về kỹ thuật. Điều này có nghĩa là các chỉ số như RMSE không hữu ích lắm vì chúng không trực quan.
Thay vào đó, hãy cho biết tỷ lệ phần trăm các dự đoán của mô hình của chúng ta nằm trong một phạm vi nhất định của giá trị chi phí thực tế.
Ví dụ, tỷ lệ phần trăm các dự đoán của mô hình của chúng ta (trên tập kiểm tra) nằm trong khoảng $2000 so với giá trị chi phí thực tế là:

```
calc_preds_in_residual_range(
    y_true=y_test,
    y_pred=y_pred_test_xgb,
    range_=2000
)
```

```
65.61085972850678
```

Chúng ta cũng có thể cho biết tỷ lệ phần trăm các dự đoán của mô hình của chúng ta nằm trong một tỷ lệ phần trăm nhất định của giá trị chi phí thực tế.
Ví dụ, tỷ lệ phần trăm các dự đoán của mô hình của chúng ta (trên tập kiểm tra) nằm trong khoảng 20% so với giá trị chi phí thực tế là:

```
calc_preds_in_residual_perc_range(
    y_true=y_test,
    y_pred=y_pred_test_xgb,
    perc_range=20
)
projectpro.checkpoint('1e808c')
```

```
53.39366515837104
```

6. CẢI TIẾN TRONG TƯƠNG LAI

Giới hạn:

- Tập dữ liệu hạn chế: Dự án này sử dụng tập dữ liệu tương đối nhỏ ( ). Việc sử dụng tập dữ liệu lớn hơn, đa dạng hơn sẽ cải thiện độ chính xác và tính tổng quát của model.insurance.csv
- Chức năng hạn chế: Tập dữ liệu chỉ chứa một số đặc điểm cơ bản về thông tin cá nhân. Việc bổ sung thêm các tính năng liên quan như tiền sử bệnh, nghề nghiệp, lối sống, v.v. có thể giúp mô hình dự đoán trở nên chính xác hơn.
- Giả định đơn giản: Mô hình tuyến tính cơ bản giả định mối quan hệ tuyến tính giữa các biến. Trên thực tế, mối quan hệ giữa chi phí y tế và các yếu tố ảnh hưởng có thể phức tạp hơn. Độ chính xác có thể được cải thiện bằng cách sử dụng các mô hình phi tuyến.
- Tối ưu hóa siêu tham số: Quá trình tối ưu hóa siêu tham số của XGBoost có thể tốn nhiều thời gian và tính toán chuyên sâu. Các kỹ thuật tối ưu hóa hiệu quả hơn có thể được sử dụng để giảm thời gian đào tạo.
- Khả năng diễn giải: Mặc dù XGBoost thường cung cấp hiệu suất dự đoán cao nhưng nó có thể khó diễn giải hơn các mô hình tuyến tính. Sử dụng kỹ thuật diễn giải mô hình có thể giúp hiểu rõ hơn cách mô hình đưa ra dự đoán.

Những cải tiến trong tương lai:

- Sử dụng tập dữ liệu lớn hơn, đa dạng hơn.
- Thêm các chức năng liên quan.
- Khám phá các mô hình phi tuyến phức tạp hơn như mạng lưới thần kinh.
- Áp dụng các kỹ thuật tối ưu hóa siêu tham số hiệu quả hơn.
- Sử dụng các kỹ thuật diễn giải mô hình để hiểu rõ hơn cách hoạt động của mô hình.
- Triển khai mô hình vào ứng dụng web hoặc API nơi người dùng có thể sử dụng mô hình đó.
- Thực hiện phân tích độ nhạy để đánh giá tác động của các tính năng và siêu tham số đến hiệu suất của mô hình.
- So sánh hiệu suất của XGBoost với các thuật toán học máy khác như rừng ngẫu nhiên, máy vectơ hỗ trợ, v.v.

7. LỜI KẾT

Dự án này mang lại giá trị thực tế cho các công ty bảo hiểm bằng cách áp dụng mô hình XGBoost để dự đoán chính xác chi phí y tế, từ đó cải thiện khả năng ước tính phí bảo hiểm. Việc so sánh giữa XGBoost và hồi quy tuyến tính cho thấy hiệu quả của các phương pháp hiện đại trong việc xử lý dữ liệu phức tạp, góp phần tăng cường tính chính xác và hiệu quả trong ngành bảo hiểm.