

Dự đoán chi phí y tế cá nhân

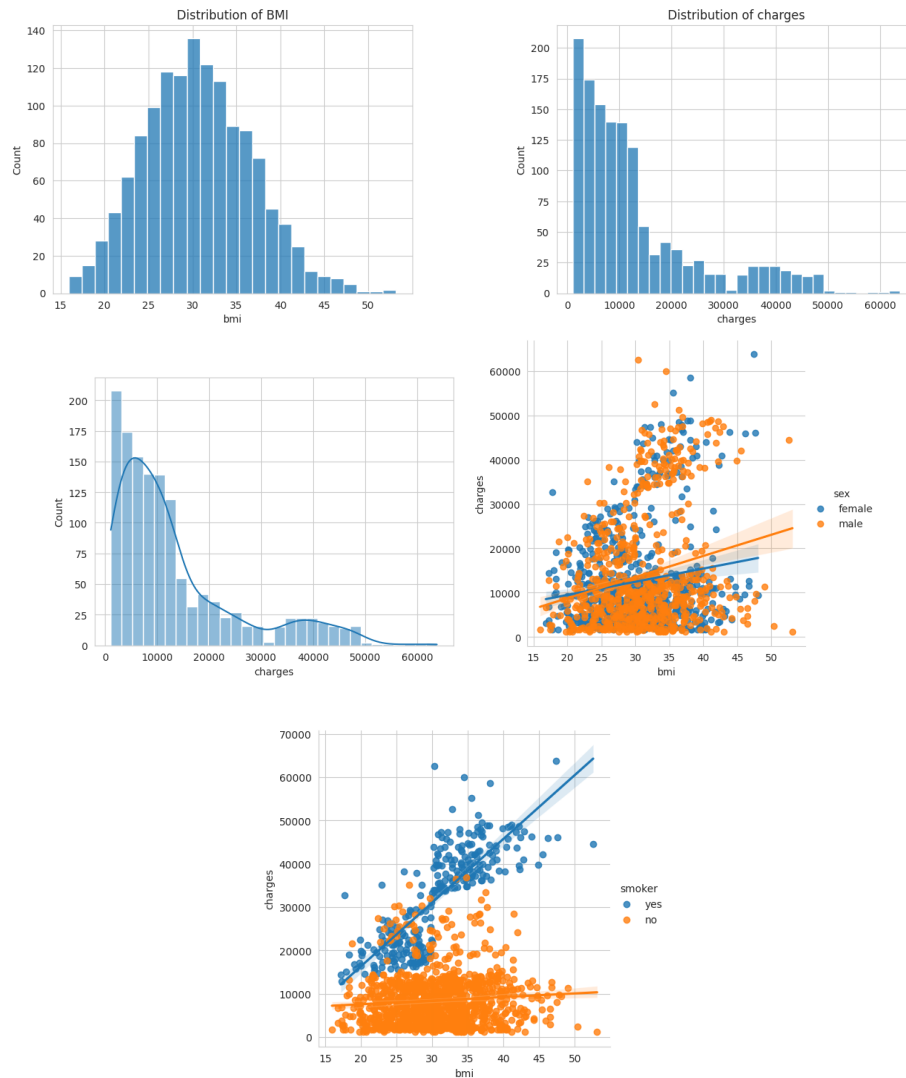
Ngày 30 tháng 6 năm 2025

0.1 Đề tài lựa chọn và các thông tin về dữ liệu

0.1.1 Đề tài

Qua các tập dữ liệu được gợi ý và các kiến thức trong khóa học về ML của VietAI, em đã suy nghĩ và quyết định chọn bộ dữ liệu MedicalCostPersonal (Chi phí y tế cá nhân). Đây là một bộ dữ liệu y tế nhỏ, với những đặc trưng quen thuộc như tuổi, chỉ số BMI,... phù hợp cho việc tập trung vào quy trình pipeline

0.1.2 Các thông tin về dữ liệu



plots

Phân bố chi phí (charges) (Biểu đồ histogram + KDE)

- Phân bố chi phí lệch phải rõ rệt.
- Phần lớn chi phí tập trung dưới 20,000.

- Có một số ít cá nhân có chi phí rất cao (trên 40,000 đến 60,000), có thể là do bệnh nặng hoặc hút thuốc.

- Đường KDE cho thấy 2 đỉnh phụ, gợi ý có thể có 2 nhóm người khác nhau về chi phí (có thể là hút thuốc và không hút thuốc).

Phân bố chỉ số BMI (Biểu đồ histogram của BMI)

- Gần như là phân bố chuẩn (phân bố hình chuông).

- Đỉnh rơi vào khoảng 30 (gần mức béo phì theo quy ước về chỉ số BMI).

- Có một số trường hợp ngoại lệ BMI > 45 cần xem xét riêng.

Phân bố chỉ số chi phí (charges) (Biểu đồ histogram của charges)

- Đa số tập trung ở phần thấp (dưới 30000).

- Nhóm chi phí cao (≥ 40000) vẫn tồn tại nhưng không phổ biến.

Mối quan hệ giữa bmi và chi phí (charges) theo giới tính (sex)

- Biểu đồ phân tán có phân biệt giới tính.

- Cả nam và nữ đều có xu hướng: BMI cao hơn \Rightarrow chi phí cao hơn.

- Nam giới có xu hướng chi phí cao hơn nữ giới với cùng mức BMI.

Mối quan hệ giữa bmi và chi phí (charges) theo tình trạng hút thuốc (smoker)

- Biểu đồ phân tán có phân biệt người hút thuốc.

- Người hút thuốc có xu hướng chi phí tăng rất nhanh theo BMI (hệ số dốc cao).

- Người không hút thuốc có chi phí tăng rất nhẹ theo BMI.

\Rightarrow Điều này cho thấy hút thuốc là yếu tố quan trọng làm gia tăng mạnh chi phí y tế, đặc biệt ở người có BMI cao.

0.2 Các bước trong pipeline (Xử lý dữ liệu, huấn luyện mô hình)

0.2.1 Xử lý dữ liệu

1. Phân chia dữ liệu

Trước hết ta chia dữ liệu thành 2 tập X đại diện cho các đặc điểm cần được cung cấp để đưa ra dự đoán và tập y đại diện cho dự đoán của model

Sau đó tiếp tục chia 2 tập X, y thành các tập train và validation nhằm huấn luyện và kiểm định

2. Thêm các đặc trưng mới cho dữ liệu (Feature engineering) Để tăng độ đa dạng cho bộ dữ liệu, giúp các mô hình học máy học được nhiều loại đặc trưng hơn, đa dạng hóa các dữ liệu đầu vào, chúng ta thêm các đặc trưng mới dựa trên những mối liên hệ trên. Các loại đặc trưng mới được tạo ra bao gồm :

- Age_bmi : Thể hiện mối quan hệ giữa tuổi và bmi bằng cách lấy cột `bmi * cột age`

- Age_smoker : Cho biết tình trạng hút thuốc thường dao động ở độ tuổi nào bằng cách lấy cột `age * cột smoker` (cột `smoker` được encode thành 0 và 1 tương ứng câu trả lời Yes và No)

Pseudocode :

```
def feature_engineering(X):  
    X = X.copy()  
    X['age_bmi'] = X['age'] * X['bmi']  
    X['age_smoker'] = X['age'] * X['smoker'].map({'yes': 1, 'no': 0})  
    return X
```

3. Tạo pipeline để xử lý các cột dữ liệu dạng categorical và các cột có dữ liệu bị thiếu

Trước hết, ta tạo một pipeline để xử lý các cột có dạng categorical (dataset này không có cột dữ liệu bị thiếu, tuy nhiên khi test, vẫn có khả năng xuất hiện các cột có dữ liệu bị khuyết)

Các phương pháp xử lý bao gồm :

- Xử lý các cột dữ liệu dạng categorical bằng One-Hot Encoder vì các cột dữ liệu cần xử lý như giới tính, tình trạng hút thuốc, vùng khảo sát là các cột

dữ liệu mang tính ngẫu nhiên, không theo thứ tự hay cấp bậc

- Sau đó điền thiếu (nếu có) các giá trị trong các cột value này bằng SimpleImputer theo quy tắc lấy các giá trị có tần số xuất hiện nhiều nhất

- Cuối cùng xử lý điền thiếu các cột dữ liệu dạng số bằng SimpleImputer (nếu có) và trả về preprocessor lưu pipeline xử lý

Pseudocode :

```
def create_preprocessing_pipeline(categorical, numerical):
    categorical_transformer = Pipeline(steps=[
        ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False)),
        ('imputer', SimpleImputer(strategy='most_frequent'))
    ])

    numerical_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='mean'))
    ])

    preprocessor = ColumnTransformer(
        transformers=[
            ('cat', categorical_transformer, categorical),
            ('num', numerical_transformer, numerical)
        ]
    )
    return preprocessor
```

0.2.2 Huấn luyện mô hình

1. LinearRegression

Mô hình đầu tiên được huấn luyện sẽ là mô hình hồi quy tuyến tính(LinearRegression), 1 mô hình cơ bản được sử dụng trong các bài toán hồi quy

Tiến hành huấn luyện mô hình với dữ liệu train (X_{train} , y_{train}) và đánh giá mô hình dựa trên tập validation với các chỉ số MAE, MSE, RMSE và R^2_{score}

Pseudocode

```
def model_score(X_val, y_val, model):
```

```

y_pred = model.predict(X_val)
mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_val, y_pred)

print(f"MAE:{mae:.2f}")
print(f"MSE:{mse:.2f}")
print(f"RMSE:{rmse:.2f}")
print(f"R2:{r2:.2f}")

def linearRegression_train():
    model = LinearRegression()
    model.fit(X_train, y_train)
    model_score(X_val, y_val, model)

```

```

➡ MAE: 4170.73
MSE: 33795080.51
RMSE: 5813.35
R2: 0.78

```

Result

Qua kết quả trên ta thấy được rằng :

- MAE: cho thấy trung bình dự đoán sẽ có độ sai lệch rơi vào khoảng 4100 so với thực tế
- MSE: cho thấy trung bình bình phương của sự khác biệt rơi vào khoảng 33 triệu giữa giá trị dự đoán của mô hình và giá trị thực tế, nhạy cảm với các giá trị ngoại lệ do sử dụng giá trị bình phương
- RMSE : là căn bậc 2 của mse, do nhấn mạnh các lỗi lớn (qua bình phương), RMSE thường lớn hơn MAE và cung cấp cái nhìn kỹ hơn về độ lệch nghiêm trọng.
- R2 score : cho thấy mô hình giải thích được khoảng 78% phương sai của giá trị thực tế, một dấu hiệu cho thấy mô hình được fit tốt

2. TensorFlow Keras (Sequential API)

Tiếp theo ta tiến hành huấn luyện mô hình Keras của TensorFlow với API Sequential

Đây là một mô hình Neural Network với các lớp neuron là được sử dụng

trong task lần này là Dense được liên kết với nhau. Như các mô hình NN khác, keras cũng được huấn luyện và học thông qua quá trình lan truyền ngược (backpropagation)

Đầu tiên ta đến với phần xây dựng mô hình tf.keras (Sequential API) với 3 hidden layers

Pseudocode :

```
def keras_model(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(input_shape,)),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.25),
        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dropout(0.25),
        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.25),

        tf.keras.layers.Dense(1)
    ])

return model
```

Sau khi đã xây dựng mô hình, ta tiến hành huấn luyện :

- Ta sử dụng optimizer là thuật toán Adam giúp điều chỉnh learning rate của model trong quá trình huấn luyện

- Với hàm mất mát (loss function), ta sử dụng Huber function, là một hàm mất mát kết hợp các điểm mạnh của MAE và MSE, giúp mô hình hoạt động hiệu quả trước các giá trị ngoại lệ (outlier)

- Để tránh tình trạng overfitting, ta thêm EarlyStopping với patience = 10 giúp ngừng quá trình huấn luyện nếu không có sự cải thiện trong loss sau 10 training epoch, với min_delta bằng 1e-4 với chỉ số cải thiện tối thiểu là min_delta

- Ngoài ra, ta có thể dùng hàm summary và lưu quá trình train của model vào 1 biến history nhằm có cái nhìn tổng quan hơn về quá trình huấn luyện

Pseudocode :

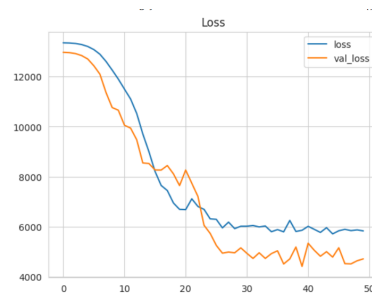
```
def fit_keras_model(X_val, y_val):
    model = keras_model(input_shape=INPUT_SHAPE)
    model.compile(optimizer='Adam', loss=Huber(), metrics=['mae', 'mse'])
    early_stopping = EarlyStopping(monitor='val_loss',
                                    patience=10,
                                    min_delta=1e-4,
                                    restore_best_weights=True)

    history = model.fit(X_train, y_train,
                        validation_data=(X_val, y_val),
                        epochs=100,
                        batch_size=16,
                        callbacks=[early_stopping])

    model.summary()
    # Fitting stats
    history_df = pd.DataFrame(history.history)
    history_df.loc[:, ['loss', 'val_loss']].plot(title="Loss")
    history_df.loc[:, ['mae', 'val_mae']].plot(title="MAE")

return model
```

- Kết quả training của model :



- Kết quả của model trên tập test :

MAE: 4418.86
MSE: 53056790.34
RMSE: 7284.01
R2: 0.66

Result

- Từ kết quả trên, ta thấy được rằng mô hình keras của tensorflow có chỉ số MAE xấp xỉ mô hình LinearRegression, tuy nhiên các chỉ số khác như MSE hay R2 score lại không tốt bằng.

=> Việc sử dụng các mô hình Multilayer Perceptron hay Neural Network trong các bài toán không mang lại hiệu quả cao do yêu cầu dữ liệu lớn và các mô hình này thường nhạy cảm với các giá trị ngoại lệ khiến cho kết quả dự đoán chịu ảnh hưởng lớn từ các giá trị ấy. **3. Tree-based model XGBoost** XGBoost là một thuật toán tree-based được tối ưu hóa từ kỹ thuật Gradient Boosting, thường được ứng dụng trong các bài toán Machine Learning nhờ hiệu suất cao (về tốc độ và độ chính xác), khả năng xử lý tốt các dữ liệu bị thiếu và các tập dữ liệu thiếu cân bằng, ... và nhiều điểm mạnh khác

Trước hết, ta khởi tạo mô hình XGBoost và tiến hành huấn luyện như mô hình LinearRegression:

Pseudocode :

```
def train_xgb():  
    xgb_model = XGBRegressor()  
    xgb_model.fit(X_train, y_train)
```

Sau đó ta đánh giá mô hình trên tập validation bằng hàm model_score ở trên

Kết quả của mô hình XGBoost :

MAE: 2377.62
MSE: 20371066.34
RMSE: 4513.43
R2: 0.87

Qua kết quả trên, ta thấy được sự vượt trội của mô hình XGBoost khi so

sánh với 2 mô hình còn lại

=> Điều đó cho thấy được rằng các mô hình tree-based như XGBoost hoạt động đặc biệt tốt trước các task về Machine Learning

0.3 Triển khai mô hình

1. Giới thiệu chung

Mô hình dự đoán chi phí y tế cá nhân được triển khai dưới dạng giao diện web trên HuggingFaceSpace, mô hình được thiết kế để dự đoán chi phí chăm sóc sức khỏe cá nhân, dựa trên các đặc trưng như: tuổi, giới tính, cân nặng, chiều cao, khu vực sinh sống và tình trạng hút thuốc. Người dùng nhập các thông tin này vào giao diện web và mô hình trả về ước tính chi phí y tế hàng năm dưới dạng khoảng giá trị trung bình.

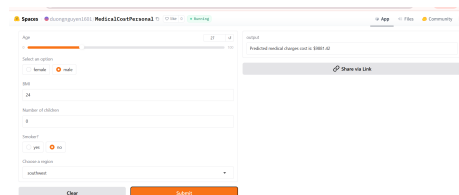
2. Cách truy cập và sử dụng

- Bước 1: Truy cập vào địa chỉ Spaces :

<https://huggingface.co/spaces/duongnguyen1601/MedicalCostPersonal>

- Bước 2: Nhập các thông tin hiển thị trên web

- Bước 3: Chọn "submit" và kết quả sẽ được hiển thị bên phần output



Hình 1: Ví dụ minh họa

0.4 Lời kết và hướng phát triển

0.4.1 Lời kết

Qua dự án và khóa học này của team VietAI, em đã học được các kĩ thuật và pipeline huấn luyện và triển khai các mô hình học máy cho các task cơ bản như dự án trên.

0.4.2 Hướng phát triển

Mặc dù các mô hình được sử dụng trong dự án lần này đã khá hoàn thiện về mặt dự đoán với mức sai số chấp nhận được, tuy nhiên, ta vẫn có thể cải thiện các giá trị dự đoán với độ sai lệch nhỏ hơn bằng các phương pháp như K-Fold (giúp cho mô hình học các feature một cách tổng quát hơn), sử dụng synthetic data (data được sinh bằng các mô hình generative AI hay deeplearning models để tăng cường dữ liệu)

Về phần triển khai, có thể triển khai mô hình trên các giao diện web tốt hơn với phần front-end được thiết kế bằng các framework phổ biến như React, Vue.js, Angular,... cùng với phần back-end sử dụng các framework như Django, Flask, ... giúp tăng trải nghiệm người dùng khi sử dụng mô hình dự đoán

Với các phương pháp kể trên, ta có thể triển khai mô hình học máy với độ chính xác và sai lệch thấp hơn với giao diện hỗ trợ người dùng tốt hơn.