PHÂN LOẠI VÀ DỰ BÁO SỐ TIỀN TIPS TAXI TẠI CHICAGO

Nguyễn Quang Thuận[1,2], Võ Trung Hiếu[1,2],
Trương Thanh Thiên[1,2], Đỗ Trọng Hợp[1,2]

[1] Trường Đại học Công nghệ Thông tin
[2] Đại học Quốc gia Thành phố Hồ Chí Minh
{18521470,18520758, 18521431}@gm.uit.edu.vn,
{hopdt}@uit.edu.vn,

Tóm tắt nội dung Để có thể nâng cao lợi nhuận của các tài xế trong cùng một khoảng thời gian làm việc thì tiền *tips* là một yếu tố quan trọng. Tuy nhiên, việc để nhận được tiền *tips* phù hợp vào nhiều yếu tố của chuyến đi như địa điểm đón/trả khách, thời gian hoặc quãng đường di chuyển. Việc hiểu được các yếu tố này có thể giúp các tài xế tối ưu lợi nhuận của chính bản thân. Ở đây, chúng tôi giúp các tài xế biết được liệu các chuyến đi có cơ hội nhận được tiền *tips* hay không? Kèm theo đó chúng tôi cũng giúp tài xế biết được số tiền *tips* có thể nhận được sau khi thực hiện chuyến đi. Từ đó các tài xế có thể tìm ra chiến lược phù hợp cho chính bản thân.

Keywords: Apache Spark - Binary Classification - Regression - Taxi Tips

1 Giới thiệu

Ngày này, nhu cầu đi lại của con người ngày càng nhiều nhưng không phải ai cũng có xe và vì thế taxi, xe ôm hay các loại hình chở hành khách khác nghiễm nhiên trở thành một mấu chốt không thể thiếu trong xã hội hiện đại. Sau khi hoàn thành một chuyến di chuyển, tài xế taxi sẽ có cơ hội nhận được tiền *tips* tùy theo các thuộc tính có trong bộ dữ liệu như quãng đường, thời gian di chuyển, tọa độ điểm đoán/trả khách và các yếu tố khác. Việc phân loại việc có nhận được *tips* cho chuyến đi đó không và nếu có thì là bao nhiêu sẽ giúp cho tài xế tăng thêm một khoảng doanh thu không ít. Trong bài báo này, chúng tôi sẽ tiến hành xây dựng hai mô hình dự đoán liệu khách hàng có *tips* sau chuyến đi không và nếu có thì là bao nhiêu? Từ đó các tài xế có thể tập trung tại một vài chỗ có tỉ lệ nhận được tiền *tips* cao để gia tăng doanh thu. Nội dung của bài báo được tóm tắt như sau. Đầu tiên chúng tôi sẽ giới thiệu nội dung bài toán, mục đích của việc thực hiện. Kế tiếp là giới thiệu bộ dữ liệu. Phần 3 chúng tôi sẽ trình bày chi tiết các phương pháp dùng để thăm dò, xử lý dữ liệu cũng như là các mô hình chúng tôi chọn để thực hiện huấn luyện và thực nghiệm trên bộ dữ liệu. Phần 4 là phần thực nghiệm sẽ nêu rõ quy trình, kết quả thực hiện từng

phương pháp, sau đó là đánh giá tất cả các kết quả có được. Cuối cùng là kết luận và đưa ra lời nhận xét cho bài báo.

2 Bộ dữ liệu

Trong bài báo này, chúng tôi sử dụng một phần của bộ dữ liệu Taxi Trip với nhiệm vụ dự đoán sau mỗi chuyến taxi thì có nhận được tiền *tips* hay không từ đó thêm mới một thuộc tính *Have Tips*. Nếu có nhận được tiền *tips* thì dự đoán số tiền *tips* nhận được là bao nhiêu dựa vào thông tin được thu thập trước đó. Các thông tin này được lấy từ bộ dữ liệu *Taxi Trips* [8] của *Jonathan Levy* trên `https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew` làm việc tại *Department of Business Affairs and Consumer Protection* của thành phố thuộc *Chicago*.

Tiền *tips* là một khoản tiền nhỏ mà khách hàng thưởng thêm cho người phục vụ nhằm thể hiện sự hài lòng cùng lời biết ơn của họ về chất lượng dịch vụ. Tiền *tips* cũng có ý nghĩa tạo động lực, niềm vui cho người phục vụ vì được khách hàng tin yêu, nhìn nhận. Tại một số quốc gia trên thế giới, tiền *tips* trở thành văn hóa; là một chuyện rất tự nhiên trong đời sống mà người được phục vụ phải thực hiện đối với người phục vụ sau khi sử dụng dịch vụ và cảm thấy hài lòng. Việc xây dựng một mô hình để dự đoán liệu chuyến đi có nhận được tiền *tips* hay không là hữu ích đối với công ty. Vì từ đó, công ty có thể lập kế hoạch chiến lược để có những chuyến taxi chất lượng và tối ưu hóa mô hình kinh doanh và doanh thu của mình.

Dữ liệu huấn luyện được cung cấp theo định dạng CSV được công bố ngày 28/05/2016 và được cập nhật gần nhất vào ngày 08/07/2021. Bộ dữ liệu được lấy tại thời điểm 23/11/2020 chứa 3.597.307 dòng và 23 cột. Trong bài báo này, chúng tôi chỉ sử dụng 500.000 dòng dữ liệu và 23 cột, đồng thời thêm mới cột *Have Tips*. Ý nghĩa của từng thuộc tính được thể hiện ở bảng 1. Trong bảng 1 bên dưới, thuộc tính *Trip ID* là mã dùng để phân biệt mỗi chuyến đi. Thuộc tính *Have Tips* thể hiện chuyến đi có được tiền *tips* hay không. 22 thuộc tính còn lại chứa thông tin của chuyến đi và từ những thông tin đó có thể rút trích được một tập đặc trưng có thể xác định được những đối tượng có đặc điểm nào sẽ ảnh hưởng tới tiền *tips* của chuyến đi. Từ đó, ta có thể dự đoán được những chuyến đi nào sẽ có tiền *tips* và vào khoảng bao nhiêu để tối ưu hóa và mang lại nhiều lợi nhuận nhất.

3 Phương pháp

3.1 Phân tích khám phá dữ liệu

Handle nan and invalid value Bộ dữ liệu gốc là dữ liệu thô nên có thể xuất hiện các giá trị NaN, giá trị không hợp lệ nên trước khi đưa vào huấn luyện thì việc xử lý các giá trị này là một điều cần thiết. Các giá trị này có thể được thay đổi thành mean, max, min hay thậm chí là loại bỏ hoàn toàn giá trị này. Việc

STT	**Tên thuộc tính**	**Giải thích**
1	Trip ID	Mã số của chuyến
2	Taxi ID	Mã số taxi
3	Trip Start Timestamp	Thời gian bắt đầu chuyến đi
4	Trip End Timestamp	Thời gian kết thúc chuyến đi
5	Trip Seconds	Thời gian toàn bộ chuyến đi (s)
6	Trip Miles	Quãng đường của chuyến đi (miles)
7	Pickup Census Tract	Dân số tại điểm đón khách
8	Dropoff Census Tract	Dân số tại điểm trả khách
9	Pickup Community Area	Khu vực cộng đồng đón khách
10	Dropoff Community Area	Khu vực cộng đồng trả khách
11	Fare	Giá tiền
12	Tips	Tiền tips
13	Tolls	Phí cầu đường
14	Extras	Phí bổ sung
15	Trip Total	Tổng chi phí chuyến đi
16	Payment type	Phương thức trả tiền
17	Company	Công ty chủ quản
18	Pickup Centroid Latitude	Vĩ độ tại điểm đón
19	Pickup Centroid Longitude	Kinh độ tại điểm đón
20	Pickup Centroid Location	Vị trí điểm đón
21	Dropoff Centroid Latitude	Vĩ độ tại điểm trả khách
22	Dropoff Centroid Longitude	Kinh độ tại điểm trả khách
23	Dropoff Centroid Location	Vị trí điểm trả khách
24	Have Tips	Có tiền tips hay không 0: Không 1: Có

Bảng 1: *Ý nghĩa các thuộc tính*

lựa chọn cách thức xử lý cần phải được xem xét kỹ càng bởi vì bất kỳ sự thay đổi nào cũng sẽ ảnh hưởng đến sự phân bố hiện tại của bộ dữ liệu gốc, việc hạn chế tối đa thay đổi của sự phân bố dữ liệu là điều cần thiết của phương pháp này.

Handle outlier value Outlier chính là một trong những vấn đề gây ra việc nhiễu dữ liệu và kéo theo đó là sự hoạt động không chính xác của mô hình dự đoán. Việc xử lý các dữ liệu outlier này là một phần bắt buộc nếu muốn hạn chế việc nhiễu dữ liệu và tăng khả năng chính xác của mô hình. Ở phần này, chúng tôi chia giá trị của thuộc tính thành 3 phần là *[Low, Medium, High]*, từ đó loại bỏ các giá trị chỉ chiếm một phần rất nhỏ trong thuộc tính đó.

Handle coordinate value Các giá trị tọa độ tồn tại trong bộ dữ liệu là dữ liệu dạng số, tuy nhiên ta không thể dùng nó để đưa vào huấn luyện cho mô hình được bởi vì tọa độ không giống như một con số bình thường. Việc xem nó như một con số bình thường sẽ làm mất đi giá trị thật và đồng thời cũng gây ra sự khó khăn cho việc huấn luyện và dự đoán. Chúng tôi sẽ chia các giá trị thuộc kinh độ và vĩ độ thành 4 phần và gán cho chúng các nhãn từ *[1, 4]* tượng trưng cho 4 phần trong vùng hoạt động của hãng xe taxi.

Standard Scale Standard Scale là một phương pháp chuẩn hoá dữ liệu, nó góp phần làm giảm thiểu outlier trong dữ liệu, đồng thời khắc phục được một số phân phối lệch của một số dataset đặc thù. Trong phạm vi đề tài này, chúng tôi sử dụng Standard Scale để đưa miền giá trị của 1 vài biến về phân phối trung tâm của nó nhằm cải thiện hiệu suất của mô hình. Công thức tổng quát cho Standard Scale:

$$r = \frac{z = (x - u)}{s}$$

Trong đó u là giá trị trung bình của các mẫu huấn luyện, s là độ lệch chuẩn của các mẫu huấn luyện.

3.2 Tiền xử lý

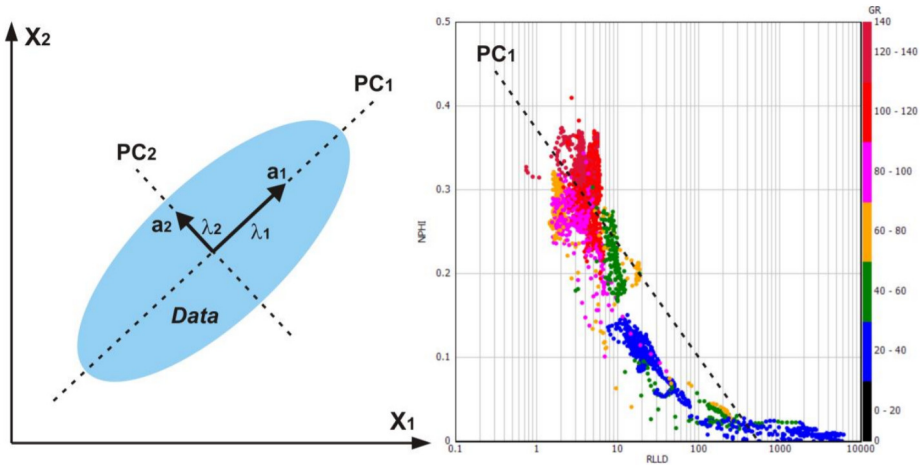
StringIndexer Dữ liệu phân loại là các biến chứa giá trị thuộc các kiểu dữ liệu bất kỳ chứ không phải giá trị số. Số lượng các giá trị có thể có thường được giới hạn trong một tập hợp cố định. Nhiều mô hình không thể hoạt động trên kiểu dữ liệu này. Chúng yêu cầu tất cả các biến đầu vào và biến đầu ra phải là số. Và để giải quyết vấn đề này thì StringIndexer là một trong những kỹ thuật Feature Engineering được sử dụng giải quyết nó trong Spark. StringIndexer sẽ mã hóa các giá trị của cột đầu vào thành dạng chỉ số. Các chỉ số nằm trong *[0, n)* với n là số lượng nhãn của cột đó. Thứ tự được sắp xếp theo tần suất xuất hiện của giá trị đó trong cột, vì vậy chỉ số của các nhãn xảy ra nhiều nhất là 0 và thấp nhất là n-1.

OneHotEncoder Đối với các biến phân loại việc tồn tại mối quan hệ có thứ tự có thể gây ra sự không chính xác đối với mô hình dự đoán. Để giải quyết vấn đề này ta có thể sử dụng kỹ thuật OneHotEncoder để đưa các con số hiện tại về giá trị vector một chiều. Việc này giúp loại bỏ mối quan hệ thứ tự đã nêu ra ở trước đồng thời vẫn giữ được tính phân loại của các giá trị, nhưng nó cũng sẽ dẫn đến việc số lượng cột mở rộng rất nhiều.

VectorAssembler VectorAssembler là một kỹ thuật trong Spark được sử dụng để gom các giá trị lại thành một vecto (features vector) trước khi đưa vào huấn luyện và dự đoán. VectorAssembler chấp nhận các kiểu cột đầu vào sau: tất cả các kiểu số, kiểu boolean và kiểu vectơ. Trong mỗi hàng, giá trị của các cột đầu vào sẽ được nối thành một vectơ theo thứ tự được chỉ định.

3.3 Lựa chọn đặc trưng và giảm chiều dữ liệu

PCA Principal Component Analysis (PCA) [1] là phương pháp giảm chiều dữ liệu và trích xuất đặc trưng tiêu biểu cho dữ liệu có cấu trúc. PCA là quá trình



Hình 1: *Hình minh họa cho phương pháp PCA [1]*

tính toán các thành phần chính và sử dụng chúng để biểu diễn sự thay đổi cơ sở trên dữ liệu, đôi khi chỉ sử dụng một số thành phần chính đầu tiên và bỏ qua phần còn lại. Quá trình thực nghiệm được chúng tôi sử dụng PCA như một phương pháp giảm chiều cho dữ liệu để từ đó đưa vào mô hình các đặc trưng quan trọng nhằm cải thiện hiệu suất mô hình.

Hệ số tương quan Pearson Hệ số tương quan của Pearson [10] là thống kê thử nghiệm đo lường mối quan hệ thống kê, hoặc sự liên kết giữa hai biến liên

tục. Nó được biết đến là phương pháp tốt nhất để đo lường mối liên hệ giữa các biến quan tâm vì nó dựa trên phương pháp hiệp phương sai. Nó cung cấp thông tin về độ lớn của mối liên kết, hoặc mối tương quan, cũng như hướng của mối quan hệ. Trong nội dung thực nghiệm đề tài, chúng tôi sử dụng hệ số tương quan này nhằm mục đích chọn ra những thuộc tính quan trọng và có ảnh hưởng cao đến biến mục tiêu, từ đó làm giảm chiều của dữ liệu trước khi đưa vào mô hình để huấn luyện. Công thức hệ số tương quan Pearson:

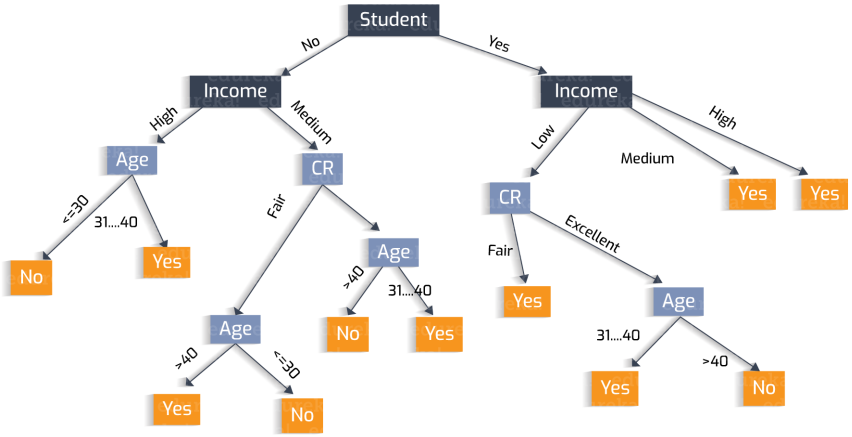
$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

3.4 Các mô hình phân lớp

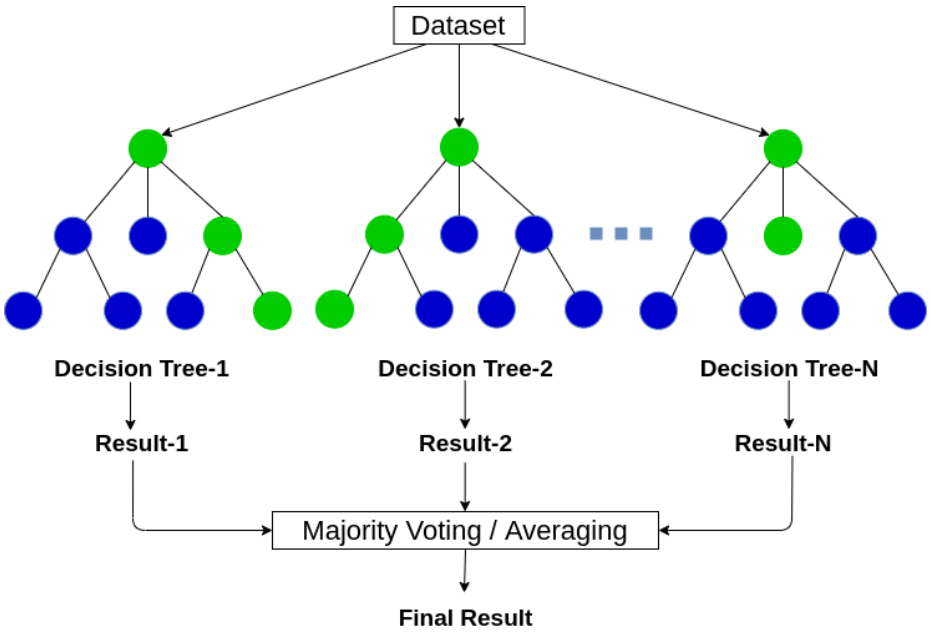
Decision Tree Decision Tree [4] là một mô hình Supervised Learning, có thể được áp dụng vào cả hai bài toán Classification và Regression. Việc xây dựng một Decision Tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của Decision Tree là nó có thể làm việc với các đặc trưng (trong các tài liệu về Decision Tree, các đặc trưng thường được gọi là thuộc tính – attribute) dạng categorical, thường là rời rạc và không có thứ tự. Ví dụ: mưa, nắng hay xanh, đỏ, v.v. Decision Tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng categorical và liên tục (numeric). Một điểm đáng lưu ý nữa là Decision Tree ít yêu cầu việc chuẩn hoá dữ liệu.

- Node quyết định (Decision Node): chỉ định kiểm tra trên một thuộc tính duy nhất.
- Nút lá (Leaf Node): cho biết giá trị của thuộc tính mục tiêu.
- Arc/Edge: tách một thuộc tính.
- Path: một sự kết hợp để tạo ra quyết định cuối cùng.

Random Forest Random Forest [11] là thuật toán Supervised Learning, có thể giải quyết cả bài toán Regression và Classification. Random là ngẫu nhiên, Forest là rừng, nên ở thuật toán Random Forest mình sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố Random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định. Trong thuật toán Decision Tree, khi xây dựng cây quyết định nếu để độ sâu tùy ý thì cây sẽ phân loại đúng hết các dữ liệu trong tập Training dẫn đến mô hình có thể dự đoán tệ trên tập Validation/Test, khi đó mô hình bị Overfitting, hay nói cách khác là mô hình có High Variance. Do mỗi cây quyết định trong thuật toán Random Forest không dùng tất cả dữ liệu Training, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị Overfitting mà có thể bị Underfitting, hay nói cách khác là mô hình có High Bias. Tuy nhiên, kết quả cuối cùng của thuật toán Random Forest lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho



Hình 2: *Hình minh họa cho mô hình Decision Tree [4]*



Hình 3: *Hình minh họa cho mô hình Random Forest [11]*

nhau, dẫn đến mô hình có Low Bias và Low Variance, hay mô hình có kết quả dự đoán tốt.

Naïve Bayes Naïve Bayes [3] là một trong họ "bộ phân loại theo xác suất" dựa trên việc áp dụng định lý Bayes trong xác suất thống kê. Với $P(y|X)$ là xác suất của mục tiêu y với điều kiện có đặc trưng X (posterior probability), $P(X|y)$ là xác suất của đặc trưng X khi đã biết mục tiêu y (likelihood), $P(y)$ là prior probability của mục tiêu y, $P(X)$ là prior probability của vector đặc trưng $X(x_1, x_2, ..., x_n)$. Khi đó,

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}[6] \tag{1}$$

Naive Bayes [3] có thể được đào tạo rất hiệu quả. Với một lần chuyển qua dữ liệu huấn luyện, nó sẽ tính toán phân phối xác suất có điều kiện của từng tính năng cho từng nhãn. Để dự đoán, nó áp dụng định lý Bayes để tính toán phân phối xác suất có điều kiện của mỗi nhãn cho một quan sát. Trong Spark, Navie Bayes có thể được đào tạo với 3 dạng là: Multinomial (mặc định), Bernoulli, Gaussian.

Support Vector Machine (SVM): SVM [9] là một mô hình Supervised Learning, có thể được áp dụng vào cả hai bài toán Classification và Regression. Margin là khoảng cách ngắn nhất từ điểm dữ liệu tới siêu phẳng phân cách giữa các lớp. Mục tiêu của bài toán là đi tìm một siêu phẳng mà ở đó margin của mỗi lớp trong bộ dữ liệu là lớn nhất và bằng nhau.

Khoảng cách từ một điểm (vector) có tọa độ $\mathbf{x_0}$ đến siêu mặt phẳng (hyperplane) có phương trình $\mathbf{w}^T\mathbf{x} + \mathbf{b} = \mathbf{0}$ được xác định bởi:

$$\frac{(\mathbf{w}^T\mathbf{x}_0 + b)}{||\mathbf{w}||_2}[7] \tag{2}$$

Với $||\mathbf{w}_2|| = \sqrt{\sum_{i=1}^{d} w_i^2}$ với d là số chiều không gian.

Margin được tính theo công thức 2; Sau đó tối ưu hóa bằng cách cập nhật lại w,b sao cho margin là lớn nhất:

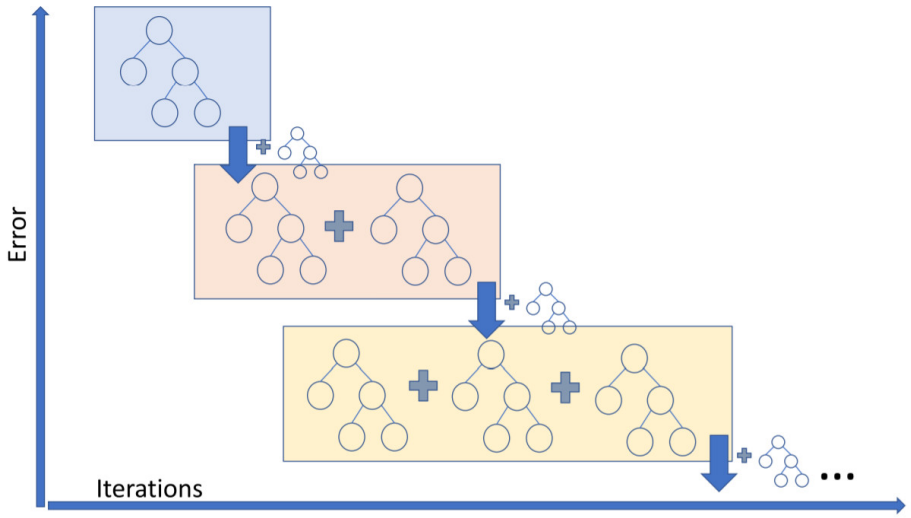
$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w^T}\mathbf{x}_n + b)}{||\mathbf{w}_2||} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}_2||} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\}[7] \tag{3}$$

3.5 Các mô hình hồi quy

Linear Regression Hồi quy tuyến tính là một mô hình tuyến tính, ví dụ: một mô hình giả định mối quan hệ tuyến tính giữa các biến đầu vào (x) và biến đầu ra duy nhất (y). Cụ thể hơn, y có thể được tính toán từ sự kết hợp tuyến tính của các biến đầu vào (x). Nói cách khác, mô hình tuyến tính sẽ là 1 siêu phẳng biểu diễn bởi các tập thuộc tính (X) với (y). Mô hình sẽ dự đoán 1 giá trị cho (y) khi đưa vào mô hình 1 tập (X) ứng với lúc huấn luyện. Công thức của mô hình hồi quy tuyến tính:

$$y = w_0 + w_1 \times x_1 + w_2 \times x_2 + \cdots + w_n \times x_n \tag{4}$$

GBTs Regression Gradient-Boosted Trees (GBTs) là một kỹ thuật máy học để hồi quy, phân loại và các tác vụ khác, tạo ra một mô hình dự đoán dưới dạng một tập hợp các mô hình dự đoán yếu, điển hình là cây quyết định. Khi một cây quyết định là người học yếu, thuật toán kết quả được gọi là cây tăng cường độ dốc, thường hoạt động tốt hơn rừng ngẫu nhiên. Nó xây dựng mô hình theo kiểu khôn ngoan theo từng giai đoạn giống như các phương pháp thúc đẩy khác và nó tổng quát hóa chúng bằng cách cho phép tối ưu hóa một hàm mất mát có thể phân biệt tùy ý.



Hình 4: *Hình minh họa cho ý tưởng mô hình GBTs Regression [2]*

Random Forest Regression Rừng ngẫu nhiên hoặc rừng quyết định ngẫu nhiên là một phương pháp học tập tổng hợp để phân loại, hồi quy và các nhiệm vụ khác hoạt động bằng cách xây dựng vô số cây quyết định tại thời điểm đào tạo. Đối với các nhiệm vụ phân loại, đầu ra của rừng ngẫu nhiên là loại được chọn bởi hầu hết các cây. Đối với các nhiệm vụ hồi quy, giá trị trung bình hoặc dự đoán trung bình của các cây riêng lẻ được trả về. Rừng quyết định ngẫu nhiên phù hợp với thói quen thích nghi quá mức của cây quyết định đối với tập huấn luyện của chúng. Rừng ngẫu nhiên nhìn chung hoạt động tốt hơn cây quyết định, nhưng độ chính xác của chúng thấp hơn cây tăng cường độ dốc (GBTRegression). Tuy nhiên, đặc điểm dữ liệu có thể ảnh hưởng đến hiệu suất của chúng.

3.6 Độ đo

Để đánh giá độ chính xác phân loại cho các mô hình đã huấn luyện, chúng tôi sử dụng độ đo Accuracy Score. Đây là độ đo thường được sử dụng để kiểm tra độ chính xác các mô hình phân loại ở hầu hết mọi bài toán.

Accuracy Accuracy là tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

$$Acc = \frac{TP+TN}{TP+FP+TN+FN}[5] \tag{5}$$

Với:

- TP (True Positive): Số điểm dữ liệu là Positive và được dự đoán là Positive
- FP (False Positive): Số điểm dữ liệu là Negative và được dự đoán là Positive
- TN (True Negative): Số điểm dữ liệu là Negative và được dự đoán là Negative
- FN (False Negative): Số điểm dữ liệu là Positive và được dự đoán là Negative

Precision Precision được định nghĩa là tỉ lệ số điểm True Positive trong số những điểm được phân loại là Positive (TP + FP).

$$Precision = \frac{TP}{TP+FP}[5] \tag{6}$$

Recall Recall được định nghĩa là tỉ lệ số điểm True Positive trong số những điểm thực sự là Positive (TP + FN).

$$Recall = \frac{TP}{TP+FN}[5] \tag{7}$$

F1-Score F1-score, là harmonic mean của Precision và Recall (giả sử rằng hai đại lượng này khác không):

$$F1 = 2\frac{Precision \cdot Recall}{Precision+Recall}[5] \tag{8}$$

RMSE Sai số toàn phương trung bình, viết tắt MSE (Mean squared error) của một phép ước lượng là trung bình của bình phương các sai số, tức là sự khác biệt giữa các ước lượng và những gì được đánh giá (giá trị thực tế). RMSE (Root mean squared error) là căn bậc hai giá trị của MSE, mục tiêu thêm căn bậc 2 là để làm giảm độ lớn của giá trị lỗi trong những trường hợp giá trị lỗi quá lớn không thể biểu diễn được. RMSE càng lớn, giá trị dự báo càng xa với giá trị thực. Nói cách khác, RMSE càng lớn, mô hình hồi quy dự báo càng tệ. Công thức tổng quát RMSE:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{d_i - f_i}{\sigma_i}\right)^2} \tag{9}$$

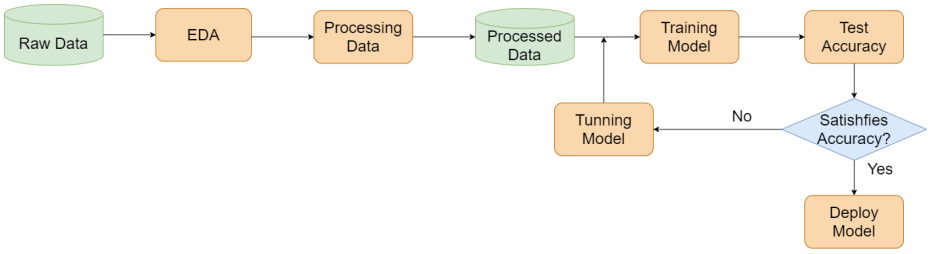
R-Squared Score R-Squared (R2) là một thước đo thống kê đại diện cho tỷ lệ phương sai của một biến phụ thuộc được giải thích bởi một biến độc lập hoặc các biến trong mô hình hồi quy. Trong khi tương quan giải thích độ mạnh của mối quan hệ giữa một biến độc lập và phụ thuộc, R2 giải thích mức độ phương sai của một biến giải thích phương sai của biến thứ hai. Vì vậy, nếu R2 của một mô hình là 0,50, thì khoảng một nửa sự thay đổi quan sát được có thể được giải thích bằng các đầu vào của mô hình. Công thức tổng quát của R2:

$$R^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}} \tag{10}$$

4 Thực nghiệm

4.1 Quy trình thực hiện

Với dữ liệu đầu vào là dữ liệu thô, chúng tôi tiến hành phân tích khám phá dữ liệu (EDA) để hiểu rõ hơn về bộ dữ liệu đang thực hiện. Qua phương pháp này chúng tôi biết được số lượng mãu trong bộ dữ liệu, các giá trị bị khuyết, không hợp lệ hay ngoại lệ, phân bố giá trị trong từng biến và cuối cùng là mối tương quan của các biến với nhau. Từ đây chúng tôi có thể chọn ra các biến có mối tương quan cao và tiến hành chuẩn hóa dữ liệu thành dạng hợp lệ để đưa vào huấn luyện mô hình. Kế tiếp chúng tôi sẽ chia dữ liệu thành hai phần là Train và Test với tỉ lệ 8/2. Lưu ý rằng chúng tôi đã cố gắng tối thiểu sự thay đổi phân phối giá trị của các biến trong quá trình EDA. Sau khi mô hình được huấn luyện, chúng tôi tiến hành thử nghiệm trên tập Test với độ đo Accuracy được nêu trên, nếu độ đo không đạt tiêu chí thì sẽ tiến hành Tuning các tham số của mô hình và quay lại bước huấn luyện mô hình.



Hình 5: *Quy trình thực hiện công việc*

4.2 Kết quả EDA

Sau khi thực hiện quá trình phân tích khám phá dữ liệu, chúng tôi thấy được số lượng mẫu của bộ dữ liệu thô là 500.000 mẫu dữ liệu và 24 thuộc tính.

Trong đó có 4 thuộc tính bao gồm *Trip Total, Pickup Centroid Location, Dropoff Centroid Location, Have Tips* là thuộc tính phụ thuộc và 19 thuộc tính còn lại là thuộc tính độc lập. Ngoài ra chúng tôi còn phát hiện có đến 260.006 mẫu dữ liệu chứa giá trị khuyết và trong đó có tới 19 thuộc tính có dữ liệu chứa giá trị khuyết trong tổng số 24 thuộc tính. Các thuộc tính thuộc nhóm thời gian, địa điểm, ID, categorical và thuộc tính *Tips* sẽ được loại bỏ bởi vì không có một căn cứ liên quan nào cho việc điền các giá trị tương đương vào các mẫu này và thuộc tính *Tips* cũng bị loại bỏ các giá trị khuyết bởi vì liên quan đến đầu ra của bài toán, nếu thay thế có thể khiến kết quả bị thay đổi. Các thuộc tính có chứa dữ liệu bị khuyết còn lại chúng tôi quyết định dùng Mean để thay thế, bởi vì sau khi tham khảo thì chúng tôi nhận thấy thay thế bằng giá trị Mean khiến cho phân phối giá trị của các thuộc tính này không bị thay đổi đi quá nhiều so với bộ dữ liệu gốc.

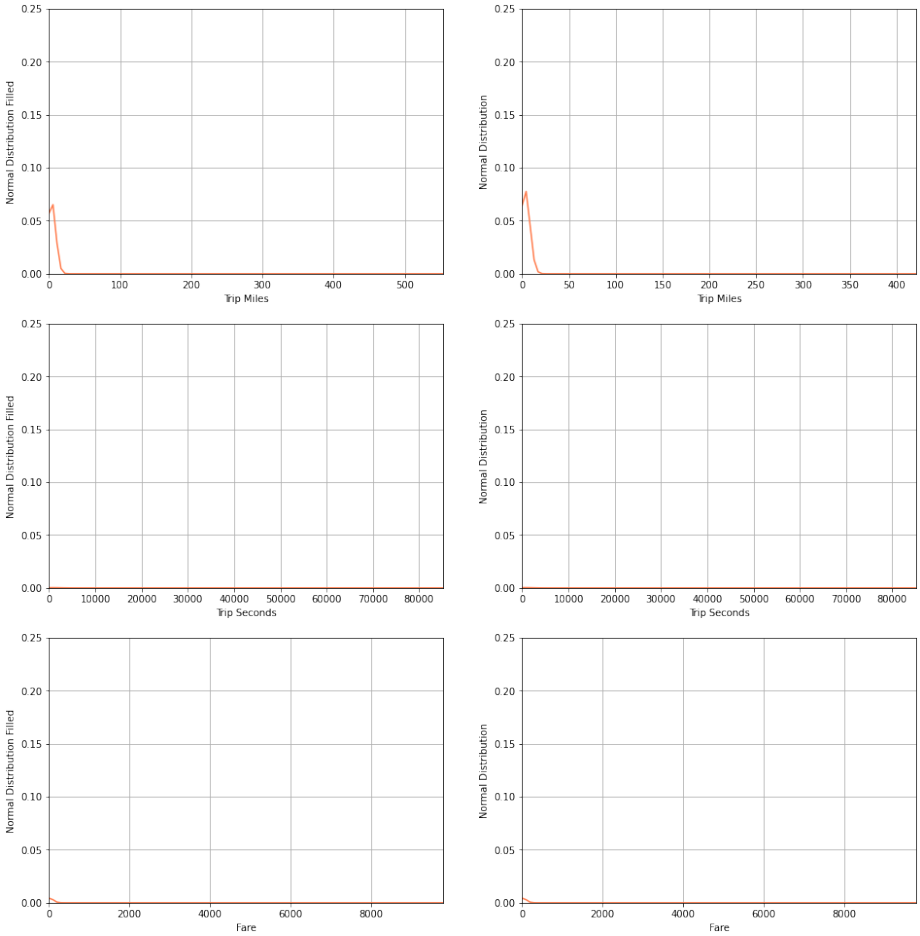
Sau đó dựa vào mối tương quan của các thuộc tính với nhau (hình 8), chúng tôi chọn ra được 7 thuộc tính có mối tương quan cao nhất với thuộc tính *Have Tips* bao gồm *Trip Miles, Pickup Census Tract, Pickup Community Area, Dropoff Community Area, Pickup Centroid Latitude, Pickup Centroid Longitude, PaymentTypeEncode*. Việc lựa chọn các thuộc tính tương quan cao để huấn luyện và thực nghiệm sẽ giúp mô hình gia tăng độ chính xác, hạn chế sự rối loạn đối với các thuộc tính ít liên quan. Kế đến chúng tôi chia giá trị tiền *Tips* theo 2 mức $0, (0, +\infty)$ (hình 7) chúng tôi nhận thấy số lượng mẫu ở 2 mức là gần tương đương nhau, không có sự chênh lệch quá nhiều. Vì để tạo thách thức cho mô hình nên chúng tôi quyết định không loại bỏ chúng.

Model	StringIndexer	OneHotEncoder	VectorAssembler	Accuracy
Decision Tree	✓	x	✓	**97.27**
Random Forest	✓	x	✓	97.14
Naive Bayes	✓	x	✓	61.11
Linear SVM	✓	x	✓	87.43
Decision Tree	✓	✓	✓	97.24
Random Forest	✓	✓	✓	97.25
Naive Bayes	✓	✓	✓	91.71
Linear SVM	✓	✓	✓	**97.27**
Decision Tree	x	x	✓	**97.27**
Random Forest	x	x	✓	97.16
Naive Bayes	x	x	✓	59.08
Linear SVM	x	x	✓	57.72

Bảng 2: *Bảng kết quả thực nghiệm với các phương pháp tiền xử lý*

4.3 Kết quả xử lý dữ liệu

Các thuộc tính tọa độ trong bộ dữ liệu những thuộc tính đầy tiềm năng, mặc dù là một con số nhưng không thể áp dụng các phương pháp xử lý thông

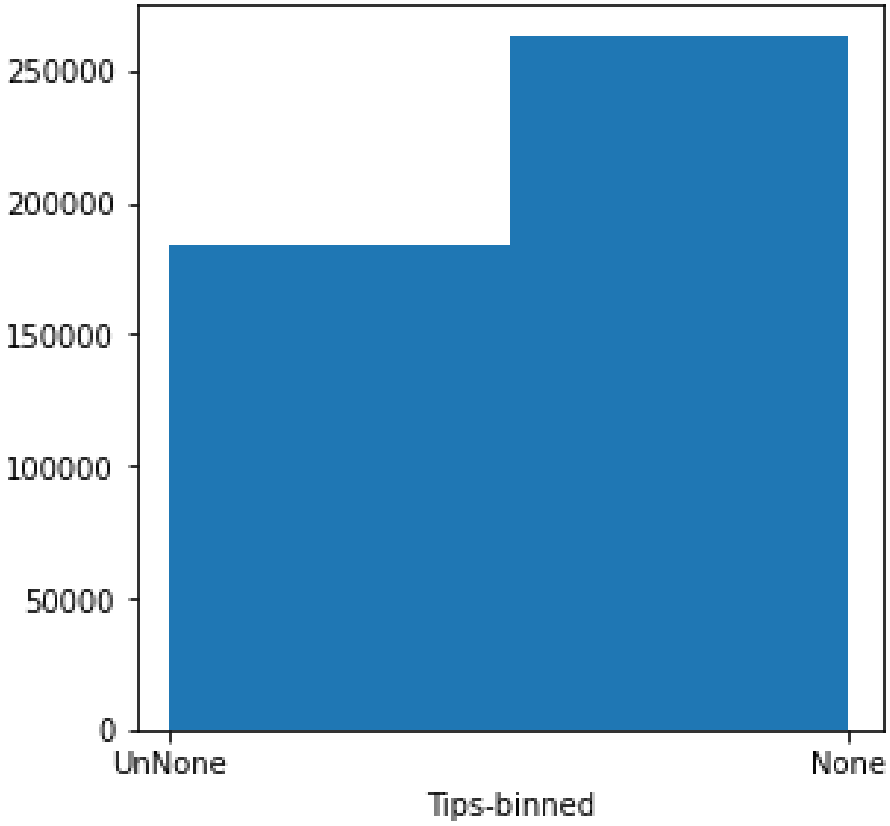


Hình 6: *Một số phân phối trước và sau khi EDA*

thường được vì chúng sẽ làm biến đổi bản chất của các thuộc tính này. Chúng tôi đã thực hiện chia các tọa độ thành 4 phần tương ứng với các mức kinh độ, vĩ độ khác nhau. Như đã nêu trên phần 3.2, chúng tôi đã sử dụng các phương pháp như StringIndexer, OneHotEncoder để chuẩn hóa dữ liệu. Và để so sánh hiệu suất của các phương pháp chúng tôi huấn luyện các mô hình phân loại với ba tùy chọn *(Dùng StringIndexer, VectorAssembler), (Dùng StringIndexer, OneHotEncoder, VectorAssembler), (VectorAssembler).*

4.4 Kết quả thực nghiệm và đánh giá các mô hình phân lớp

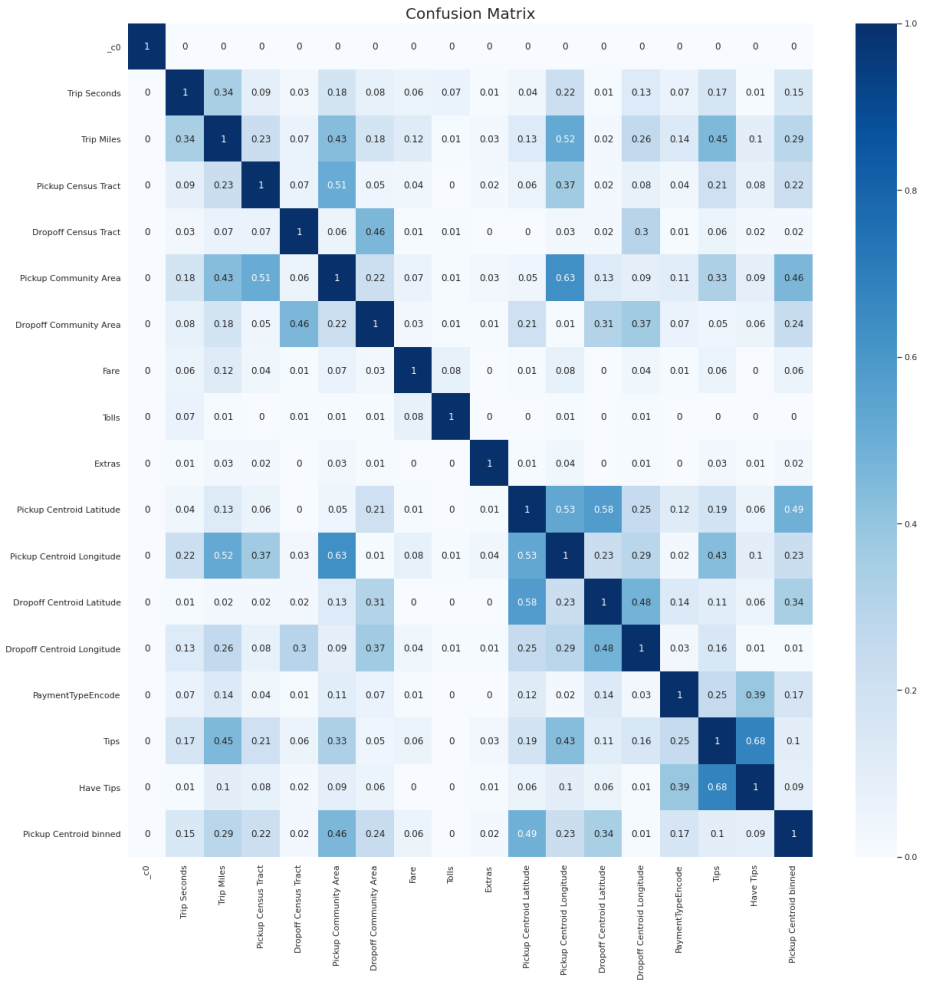
Sau quá trình phân tích và xử lý dữ liệu chúng tôi có được tập huấn luyện bao gồm 300.088 dòng và tập kiểm tra bao gồm 75.465 dòng. Đối với tất cả các



Hình 7: *Phân bố giá trị thuộc tính Tips*

thử nghiệm, chúng tôi dùng độ đo Accuracy Score để đánh giá, từ đây chọn ra mô hình đạt hiệu suất tốt nhất.

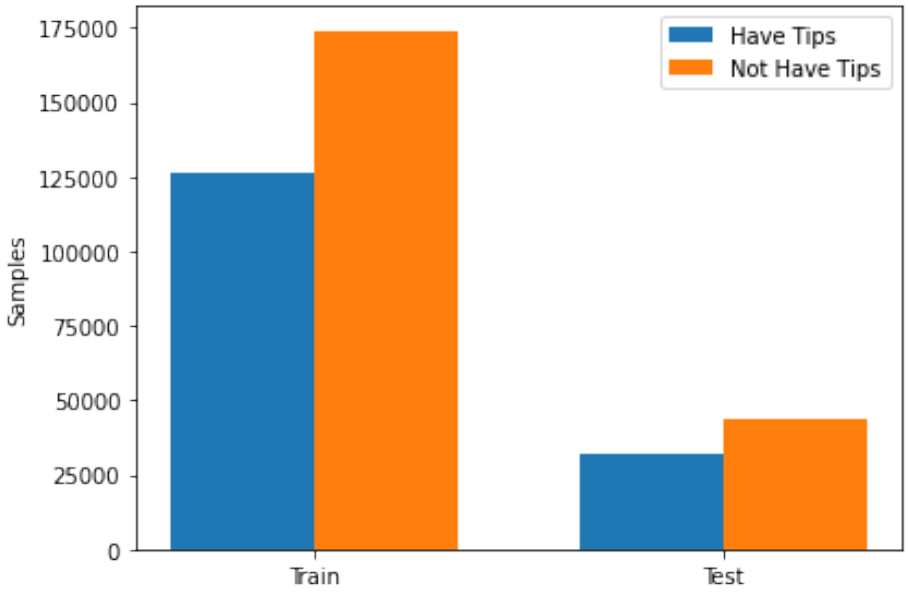
Kết quả của các mô hình kết hợp với các phương pháp tiền xử lý được thể hiện trong bảng 2. Chúng tôi đã thử nghiệm các mô hình này với các phương pháp tiền xử lý trên 7 thuộc tính có độ tương quan cao nhất với thuộc tính *Have Tips*. Nhìn vào số liệu có thể thấy được đa số các thuộc tính đưa vào huấn luyện đều thuộc kiểu dữ liệu categorical nên cả Decision Tree và Random Forest đều đạt được hiệu suất cao trên ba lựa chọn. Tuy nhiên mô hình đạt được hiệu suất cao nhất trong ba lựa chọn là LinearSVM với các phương pháp tiền xử lý StringIndexer, OneHotEncoder và VectorAssembler với Accuracy-Score đạt đến 97.2769%. Qua đó cũng chứng minh các phương pháp StringIndexer và OntHotEncoder cũng chứng minh được tính hiệu quả thông qua mô hình *Naive Bayes* khi chỉ đạt 59.08% với việc chỉ sử dụng phương pháp VectorAssembler, sau khi áp dụng thêm phương pháp StringIndexer thì tăng lên 61.11% và sau khi được áp dụng OneHotEncoder thì tỉ lệ chính xác tăng lên đến 91.71%. Qua



Hình 8: *Độ tương quan giữa các thuộc tính*

đó cho thấy việc xử lý dữ liệu đúng cách trước khi đưa vào quá trình huấn luyện sẽ giúp mô hình hoạt động tốt hơn trong lúc huấn luyện và thực nghiệm.

Kết quả thực nghiệm và đánh giá các mô hình hồi quy Chúng tôi sử dụng cùng tập train/ test với 4 thuộc tính *(['Trip Miles', 'Pickup Community Area', 'Pickup Centroid Longitude', 'PaymentTypeEncode'])* có độ tương quan cao nhất với biến mục tiêu*(Tips)* cho quá trình huấn luyện các mô hình hồi quy. Mục tiêu là có thể dự đoán được cụ thể số tiền Tips mà người tài xế đó có thể nhận nếu chuyến taxi đó được dự báo là sẽ có tiền Tips (ở mục 4.4)
Chúng tôi tiến hành đánh giá trên nhiều mô mình với nhiều cách xử lí đầu vào khác nhau (PCA, Correlation Selection, One-Hot Encoder,...) Cuối cùng chọn



Hình 9: *Thống kê số lượng nhãn ở mỗi tập*

ra được mô hình GBTs Regression cho hiệu suất cao nhất trên tập test (76.2% trong đó mô hình sử dụng với các phương pháp tiền xử lí như One-Hot Encoder, Standar Scaler.
Kết quả cuối cùng được đánh giá trên 33506 dòng dữ liệu được dự báo phân loại là có Tips (Have Tips Prediction = 1). Kết quả cao nhất thu được với độ chính xác là 66.5%. Bảng chi tiết kết quả đánh giá được chúng tôi cung cấp bên dưới.

Methods	RMSE	R2
LR with PCA (k= 9)	2.206	0.274
LR with Our Selection (n= 4) + SC	2.169	0.298
LR with Our Selection (n= 4) + OHE + SC	1.159	0.623
RF with PCA (k= 9)	1.409	0.704
RF with Our Selection (n= 4) + SC	1.376	0.717
RF with Our Selection (n= 4) + OHE + SC	1.298	0.747
GBTs with PCA (k= 9)	1.297	0.749
GBTs with Our Selection (n= 4) + SC	1.295	0.750
GBTs with Our Selection (n= 4) + OHE + SC	1.261	0.762

Bảng 3: Bảng đánh giá hiệu suất mô hình hồi quy trên tập test gốc

Methods	RMSE	R2
LR with Our Selection (n= 4) + OHE + SC	2.114	0.506
RF with Our Selection (n= 4) + OHE + SC	1.885	0.607
GBTs with Our Selection (n= 4) + OHE + SC	1.740	0.665

Bảng 4: Bảng đánh giá hiệu suất mô hình hồi quy trên các dòng dự báo cho Have Tips bằng 1

5 Kết luận

Trong bài báo cáo này, chúng tôi đã giải quyết bài toán phân loại việc tài xế taxi có nhận được tiền *Tips* khi thực hiện chuyến đi hay không và nếu có thì số tiền nhận được là bao nhiêu. Chúng tôi đã thực hiện các bước phân tích thăm dò, tiền xử lý dữ liệu, chuẩn hóa dữ liệu trước khi đưa vào quá trình huấn luyện với mục đích nâng cao kết quả. Dựa vào đó, chúng tôi đã đề xuất các phương pháp giải quyết phù hợp và đạt hiệu suất tương đối trong thực nghiệm (bảng 2) với mô hình *LinearSVM* kèm theo hiệu suất lên đến 97.2769%.

Đối với kết quả hồi quy (bảng 3) với mô hình *GBTs Regression* Đạt độ chính xác 76.2%. Mặc dù kết quả khá cao nhưng khi đánh giá trên các dòng dữ liệu được dự đoán là có tiền Tips (Have Tips Prediction = 1) lại bị giảm độ chính xác xuống còn 66.5%.

Về cơ bản, trong bài báo cáo này, chúng tôi đã giải quyết được bài toán đặt ra, tuy nhiên độ chính xác vẫn chưa cao. Hướng phát triển được chúng tôi đề xuất trong tương lai là sử dụng các mô hình tiên tiến hơn và nhiều cách xử lí dữ liệu đầu vào phức tạp hơn để từ đó làm tăng hiệu suất tổng thể của hệ thống.

Lời cảm ơn

Nhóm chúng tôi xin chân thành gửi lời cảm ơn đến TS. Đỗ Trọng Hợp - giảng viên phụ trách môn Phân tích Dữ liệu lớn, đã tận tình giúp đỡ, đưa ra những góp ý để chúng tôi hoàn thiện hơn bài báo cáo.

Tài liệu

[1] Gina Andrei and Bogdan Mihai Niculescu. “Principal component analysis as a tool for enhanced well log interpretation”. In: *Geoscience* (2016).

[2] Ivanna Baturynska and Kristian Martinsen. “Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms”. In: *Journal of Intelligent Manufacturing* 32.1 (2021), pp. 179–200.

[3] Gaurav Chauhan. *All about Naive Bayes*. Oct. 2018. URL: `https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf1`.

[4] Nagesh Singh Chauhan. *Decision Tree Algorithm, Explained*. Jan. 2020. URL: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.
[5] Long Vu Huu. *Các phương pháp đánh giá một hệ thống phân lớp*. Jan. 2018. URL: https://machinelearningcoban.com/2017/08/31/evaluation/.
[6] Long Vu Huu. *Naive Bayes Classifier*. Aug. 2017. URL: https://machinelearningcoban.com/2017/08/08/nbc/.
[7] Long Vu Huu. *Support Vector Machine*. Apr. 2017. URL: https://machinelearningcoban.com/2017/04/09/smv/.
[8] Anmol Kumar. *Taxi Trips in Chicago city*. Nov. 2020. URL: https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew.
[9] Rushikesh Pupale. *Support Vector Machines(SVM) — An Overview*. June 2018. URL: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989.
[10] Philip Sedgwick. "Pearson's correlation coefficient". In: *Bmj 345* (2012).
[11] Tony Yiu. *Understanding Random Forest*. June 2019. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.