ĐẠI HỌC UEH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH
BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÁY HỌC

CHỦ ĐỀ : CLASSIFICATION

Đề tài: PHÂN HẠNG PHÂN KHÚC GIÁ ĐIỆN THOẠI DI ĐỘNG DỰA TRÊN CÁC TÍNH NĂNG THIẾT BỊ

GVHD : TS. Đặng Ngọc Hoàng Thành

MÃ HỌC PHẦN: 23D1INF50904401

NHÓM THỰC HIỆN : 9

THÀNH VIÊN NHÓM: Nguyễn Khánh Linh
Đồng Sỹ Tú
Chu Đỗ Tài Trí
Hồ Viễn Triết

TP. Hồ Chí Minh, Tháng 4/2023

MỤC LỤC

LỜI MỞ ĐẦU

Machine learning có khởi nguồn từ lĩnh vực trí tuệ nhân tạo (AI). Đây là giải pháp có khả năng khiến các ứng dụng, phần mềm trở nên thông minh hơn mà không cần thực hiện bất kỳ mã hóa nào. Thông qua dữ liệu đầu vào, Machine learning có thể dự đoán các giá trị đầu ra mới.

Machine learning là một lĩnh vực nghiên cứu cực kỳ hứa hẹn trong khoa học máy tính và trí tuệ nhân tạo. Nó cho phép các máy tính và hệ thống thông minh tự động học hỏi từ dữ liệu và cải thiện hiệu suất của chúng theo thời gian. Các ứng dụng của Machine learning ngày càng trở nên phổ biến và đa dạng, từ xử lý ngôn ngữ tự nhiên cho đến xử lý hình ảnh, nhận dạng giọng nói, và nhiều hơn nữa.

Được xem là một phần quan trọng thuộc lĩnh vực khoa học dữ liệu. Chúng sử dụng các phương pháp thống kê, thuật toán để phân loại, dự đoán và khám phá những thông tin quan trọng của dữ liệu. Nhờ những chuỗi thông tin này, người dùng nhanh chóng đưa ra các quyết định trong các hoạt động kinh doanh của mình. Vì vậy, Machine learning chính là giải pháp lý tưởng giúp doanh nghiệp tác động đến chỉ số tăng trưởng doanh thu.

Nhờ vào các khả năng của Machine Leaning mang lại, chúng em muốn tiến hành nghiên cứu bài tiểu luận "*Phân hạng phân khúc giá điện thoại di động dựa vào các tín năng thiết bị*" nhằm mục đích áp dụng mô hình Machine learning đã học, đồng thời áp dụng nâng cao thêm mô hình Deep Learning để giải quyết bài toán thực tiễn này.

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

BẢNG PHÂN CÔNG

STT	Họ và tên	Công việc	Mức độ hoàn thành
1	Hồ Viễn Triết	- Code phần Mô hình và đánh giá chỉ số - Viết Word chương 1	100%
2	Chu Đỗ Tài Trí	- Code phần Mô hình và đánh giá chỉ số - Viết Word chương 2	100%
3	Đồng Sỹ Tú	- Code phần Tiền xử lý và Biểu đồ - Viết Word chương 3	100%
4	Nguyễn Khánh Linh	- Code phần Tiền xử lý và Biểu đồ - Viết Word chương 4	100%

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. LÝ DO CHỌN ĐỀ TÀI:

Việc đưa ra một quyết đinh mua một chiếc điện thoại đối với con người hiện nay có lẽ gây bối rối với nhiều người bởi theo thống kê, hiện nay có hơn **5 tỷ** người sử dụng điện thoại trên toàn thế giới và với sự phát triển nhanh chóng của công nghệ di động, có hàng trăm loại điện thoại được phát hành mỗi năm với đầy đủ các tính năng và các mức giá khác nhau.

Trong thực tế, người dùng thường đặt câu hỏi là nên mua điện thoại ở mức giá nào để đảm bảo chất lượng và hợp lý với ngân sách của mình. Do đó, việc phân loại giá của điện thoại di động sẽ giúp người dùng dễ dàng hơn trong việc lựa chọn sản phẩm phù hợp với nhu cầu và khả năng tài chính của mình.

Vậy nên, việc sử dụng ***Machine Learning*** để phân loại giá của điện thoại di động sẽ giúp cho các nhà sản xuất có thể đưa ra giá cả phù hợp với thị trường và đáp ứng được nhu cầu của khách hàng. Đồng thời, người tiêu dùng cũng sẽ được hỗ trợ trong việc chọn mua điện thoại.

Kết hợp giữa việc thu nhập dữ liệu về các loại điện thoại di động khác nhau và phân loại chúng thành các nhóm giá khác nhau, đề tài chúng em nghiên cứu góp phần cho việc phát triển các thuật toán máy học và trí tuệ nhân tạo để có thể giúp các doanh nghiệp sản xuất điện thoại phân loại, định giá sản phẩm của mình một cách hiệu quả và có thể đưa ra các khuyến mãi và giá cả phù hợp với mục tiêu của khách hàng.

Vì thế, đề tài "*Phân hạng phân khúc giá điện thoại di động dựa vào các tín năng thiết bị*" được nghiên cứu dựa trên bộ dữ liệu **Mobile Price Classification** là một lựa chọn hợp lý và thú vị, đồng thời mang tính ứng dụng cao trong thực tế đời sống của chúng ta.

1.2. GIỚI THIỆU BỘ DỮ LIỆU :

Tác giả bộ dữ liệu: ABHISHEK SHARMA và bộ dữ liệu được cập nhật từ năm 2018.

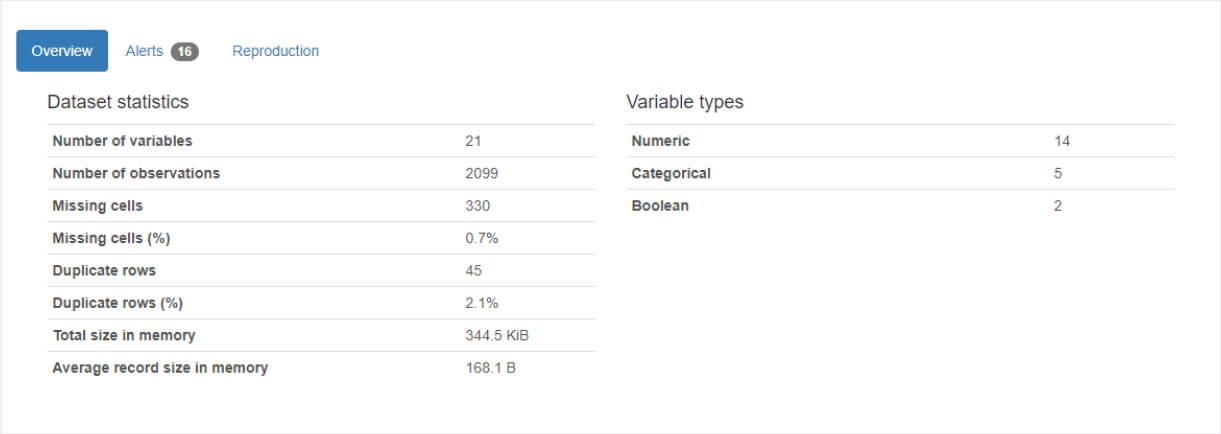
1.2.1. Case study:

" Ông Bob muốn thành lập công ty điện thoại di động cho riêng mình và ông ấy muốn tiến hành một cuộc chiến khó khăn với các thương hiệu điện thoại thông minh lớn như Apple, Samsung, ... Nhưng ông ấy không biết cách ước tính giá điện thoại di động để có thể trang trải các chi phí tiếp thị, sản xuất,...
Trong thị trường điện thoại di động cạnh tranh này, không thể đơn giản giả định mọi thứ. Để giải quyết vấn đề này, Bob thu thập dữ liệu bán điện thoại di động của nhiều công ty khác nhau. Với mục đích Bob mong muốn là tìm hiểu một số mối quan hệ giữa các tính năng của điện thoại di động và ước lượng khoảng giá bán của nó. Nhưng ông ấy không giỏi về Học máy. Vì vậy, ông ấy cần sự giúp đỡ của bạn để giải quyết vấn đề này."

Để giải quyết bài toán này, nhóm em sẽ sử dụng các mô hình máy học để ước lượng, dự đoán phạm vi giá điện thoại để xem xét chúng thuộc vào khoảng giá như thế nào.

1.2.2. Tổng quan bộ dữ liệu :

Dữ liệu này được thu nhập từ việc bán điện thoại di động từ nhiều công ty khác nhau bao gồm : *21 cột , 2099 dòng với 45 dòng dữ liệu bị trùng lặp và có 330 dữ liệu bị thiếu.*

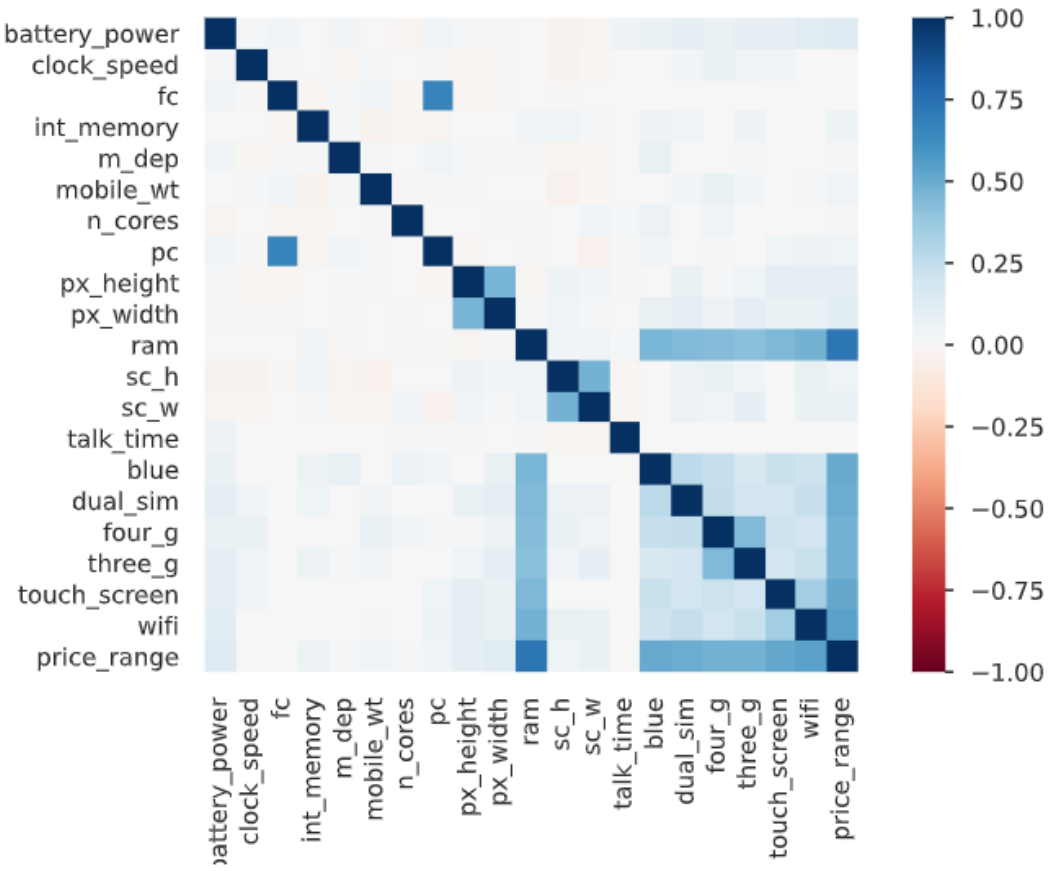


Hình 1: Tổng quan bộ dữ liệu

Bảng 1: Bảng giải thích biến trong bộ dữ liệu:

Tên biến	Giải thích biến
battery_power	Tổng năng lượng mà pin có thể lưu trữ trong một lần đo bằng mAh
blue	Có bluetooth hay không
clock_speed	tốc độ mà bộ vi xử lý thực hiện các hướng dẫn
dual_sim	Có hỗ trợ sim kép hay không
fc	Camera trước megapixel
four_g	Has 4G or not
int_memory	Bộ nhớ trong tính bằng Gigabyte
m_dep	Độ dày điện thoại tính bằng cm
mobile_wt	Trọng lượng của điện thoại di động
n_cores	Số lõi của bộ xử lý
pc	Camera chính mega pixel
px_height	Chiều cao độ phân giải pixel
px_width	Chiều rộng độ phân giải pixel

ram	Ram tính bằng Mega Byte
sc_h	Chiều cao màn hình của thiết bị di động tính bằng cm
sc_w	Chiều rộng màn hình của thiết bị di động tính bằng cm
talk_time	Thời gian dài nhất mà một lần sạc pin sẽ kéo dài
three_g	Có 3G hay không
touch_screen	Có màn hình cảm ứng hay không
wifi	Có wifi hay không
price_range *(Biến target)*	Đây là biến mục tiêu có giá trị 0 (chi phí thấp), 1 (chi phí trung bình), 2 (chi phí cao) và 3 (chi phí rất cao).



Hình 2: Heatmap thể hiện độ tương quan giữa các biến

1.3. MỤC ĐÍCH NGHIÊN CỨU :

Việc áp dụng các kỹ thuật Machine Learning đã được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như phát hiện gian lận tài khoản, phân loại ảnh, dự báo giá cả và nhiều ứng dụng

khác. Vì vậy, nghiên cứu về Mobile Price Classification không chỉ có giá trị trong lĩnh vực công nghệ thông tin mà còn có thể ứng dụng trong nhiều lĩnh vực khác nhau.

Ngoài ra, việc nghiên cứu này cũng cung cấp cho chúng ta cơ hội giúp chúng ta có một cái nhìn tổng quan về phân loại giá điện thoại, khi khách hàng nhìn vào nó sẽ giúp ta dễ dàng lựa chọn được một chiếc điện thoại mà chất lượng của nó phù hợp với túi tiền, nhu cầu và khả năng của mình

1.4. PHƯƠNG PHÁP NGHIÊN CỨU:

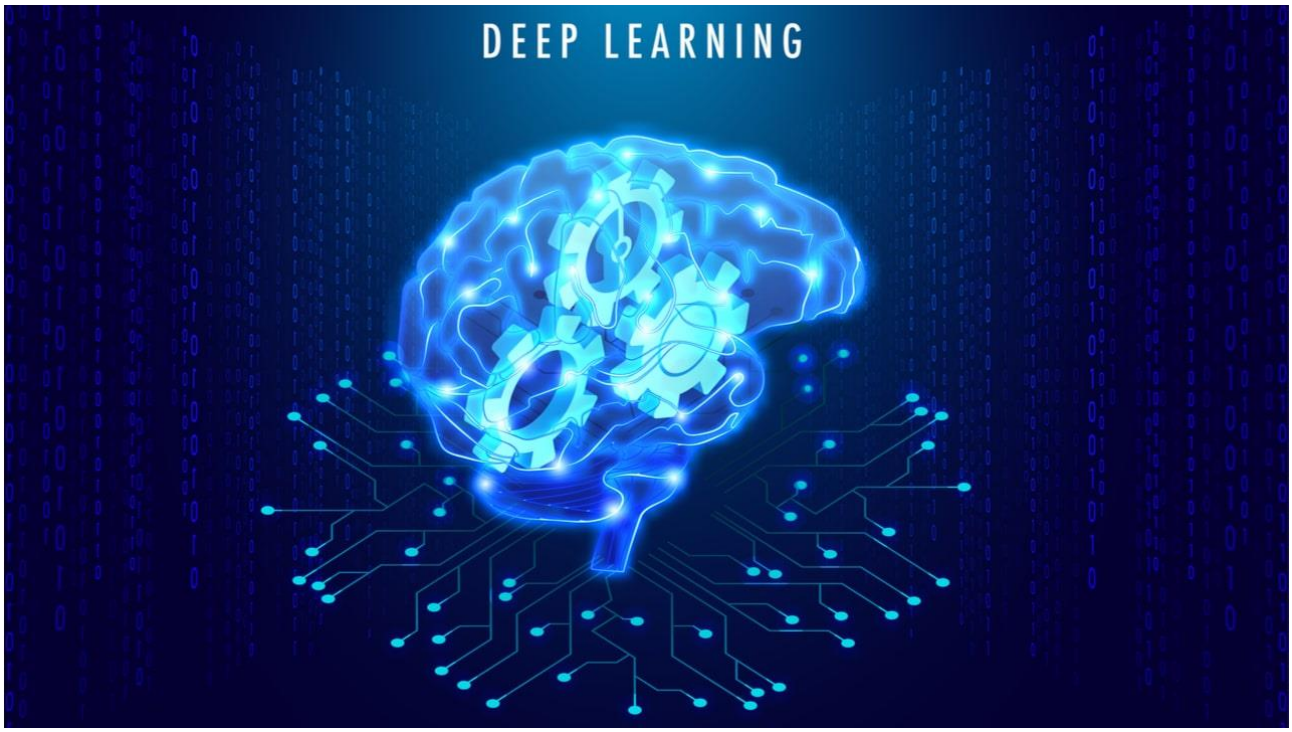
Nhóm sẽ dùng 3 mô hình để giải quyết vấn đề này, bao gồm 2 mô hình máy học là ***Random Forest, Support Vector Machies*** (SVM) và 1 mô hình học sâu là ***Multilayer Perceptron*** *(*MLP) để xây dựng mô hình phân loại giá của điện thoại di động. Sau đó sẽ đánh giá hiệu suất của mô hình bằng cách sử dụng các phương pháp đánh giá như *confusion matrix, accuracy, precision, recall và F1-score*. Kết quả đánh giá sẽ giúp chúng ta đánh giá xem mô hình của chúng ta có đạt được hiệu suất tốt hay không và điều chỉnh lại các thông số của mô hình nếu cần thiết. Từ đó, ta có thể đưa ra dự đoán về khoảng giá cả của một điện thoại di động mới dựa trên các thông số kỹ thuật được cung cấp. Ngoài ra chúng ta đồng thời phân tích được những biến nào ảnh hưởng mạnh đến việc phân loại giá của điện thoại.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. DEEP LEARNING:

2.1.1. Khái niệm về Deep Learning:

Deep Learning (học sâu) có thể được xem là một lĩnh vực con của ***Machine Learning*** (Máy học) – ở đó các máy tính sẽ học và cải thiện chính nó thông qua các thuật toán. Deep Learning được xây dựng dựa trên các khái niệm phức tạp hơn rất nhiều, chủ yếu hoạt động với các mạng nơ-ron nhân tạo để bắt chước khả năng tư duy và suy nghĩ của bộ não con người.

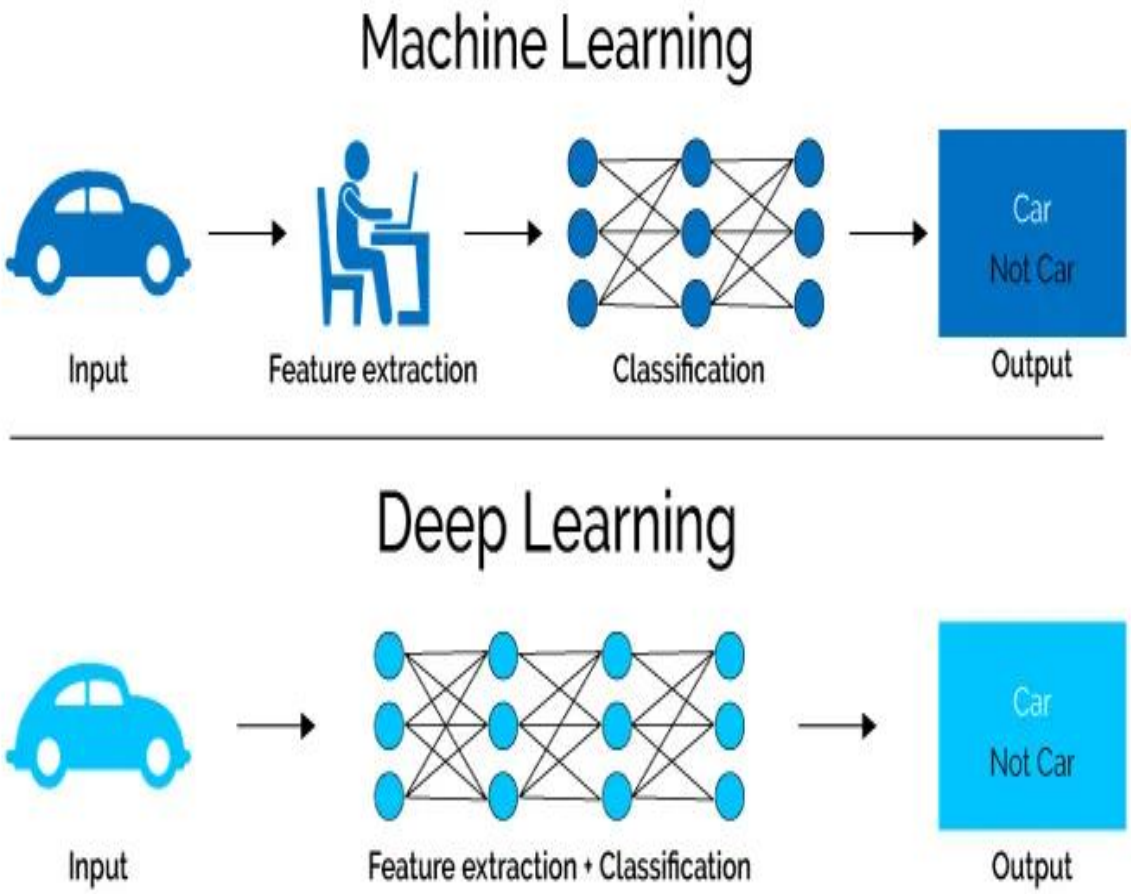


Hình 3: Hình minh hoạ DEEP LEARNING

Trong những năm gần đây, những tiến bộ trong phân tích dữ liệu lớn ***(Big Data)*** đã cho phép ta tận dụng được tối đa khả năng của mạng nơ-ron nhân tạo mà mạng nơ-ron nhân tạo chính là động lực chính để phát triển Deep Learning. Các mạng nơ-ron sâu bao gồm nhiều lớp nơ-ron khác nhau, có khả năng thực hiện các tính toán có độ phức tạp rất cao. Deep Learning hiện đang phát triển rất nhanh và được xem là một trong những bước đột phá lớn nhất trong Machine Learning

2.1.2. Deep Learning hoạt động như thế nào :

Một mạng nơ-ron bao gồm nhiều lớp (layer) khác nhau, số lượng layer càng nhiều thì mạng sẽ càng "sâu". Trong mỗi layer là các nút mạng (node) và được liên kết với những lớp liền kề khác. Mỗi kết nối giữa các node sẽ có một trọng số tương ứng, trọng số càng cao thì ảnh hưởng của kết nối này đến mạng nơ-ron càng lớn.
Mỗi nơ-ron sẽ có một hàm kích hoạt, về cơ bản thì có nhiệm vụ "chuẩn hoá" đầu ra từ nơ-ron này. Dữ liệu được người dùng đưa vào mạng nơ-ron sẽ đi qua tất cả layer và trả về kết quả ở layer cuối cùng, gọi là output layer.



Hình 4: Deep Learning hoạt động như thế nào?

2.1.3. Ưu nhược điểm của Deep Learning:

- **Ưu điểm:** Một số ưu điểm vượt trội của Deep Learning gồm có:
- Kiến trúc mạng nơ-ron linh hoạt, có thể dễ dàng thay đổi để phù hợp với nhiều vấn đề khác nhau.
- Có khả năng giải quyết nhiều bài toán phức tạp với độ chính xác rất cao.
- Tính tự động hoá cao, có khả năng tự điều chỉnh và tự tối ưu.
- Có khả năng thực hiện tính toán song song, hiệu năng tốt, xử lý được lượng dữ liệu lớn.

- **Nhược điểm:** Bên cạnh những ưu điểm, mặt khác, hiện nay Deep Learning vẫn còn nhiều khó khăn và hạn chế, chẳng hạn như:
- Cần có khối lượng dữ liệu rất lớn để tận dụng tối đa khả năng của Deep Learning.
- Chi phí tính toán cao vì phải xử lý nhiều mô hình phức tạp.
- Chưa có nền tảng lý thuyết mạnh mẽ để lựa chọn các công cụ tối ưu cho Deep Learning

2.1.4. Ứng dụng của Deep Learning:

Kiến trúc mạng nơ-ron trong Deep Learning được ứng dụng trong các công việc yêu cầu sức mạnh tính toán cao, xử lý nhiều dữ liệu và độ phức tạp lớn. Sau đây là 5 ứng dụng thân thuộc nhất của Deep Learning trong đời sống hàng ngày:

2.1.4.1. Xe tự lái :

Được xây dựng dựa trên các mạng nơ-ron cấp cao. Nói một cách đơn giản, các mô hình Deep Learning sẽ nhận diện các đối tượng ở môi trường xung quanh xe, tính toán khoảng cách giữa xe và các phương tiện khác, xác định vị trí làn đường, tín hiệu giao thông,… từ đó đưa ra được các quyết định tối ưu và nhanh chóng nhất.

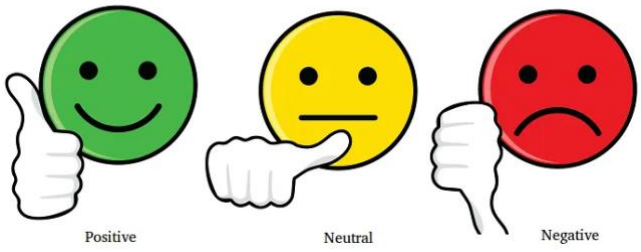
Một trong những hãng xe tiên phong trong việc sản xuất xe tự lái hiện nay là Tesla.

Hình 5: Ứng dụng xe tự lái

2.1.4.2. Phân tích cảm xúc :

Đây là lĩnh vực phân tích cảm xúc của con người thông qua việc xử lý ngôn ngữ tự nhiên, phân tích văn bản và thống kê.

Các công ty có thể ứng dụng Deep Learning để hiểu và phán đoán cảm xúc của khách hàng dựa trên những đánh giá, bình luận, tweet,… từ đó đưa ra những chiến lược kinh doanh và marketing phù hợp với từng nhóm đối tượng.



Hình 6: Phân tích cảm xúc

2.1.4.3. Trợ lý ảo :

Trợ lý ảo đang được ứng dụng rất nhiều trong đời sống hàng ngày, trong đó phổ biến gồm có chatbot, giảng viên online, Google Assistant, Siri, Cortana,…

Các trợ lý ảo được xây dựng dựa trên Deep Learning với các thuật toán nhận diện văn bản, xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói,…

Hình 7: Trợ lý ảo

2.1.4.4. Mạng xã hội :

Một số nền tảng mạng xã hội lớn như Twitter cũng ứng dụng các thuật toán Deep Learning để cải thiện các dịch vụ của mình. Cụ thể, những trang này sẽ phân tích một lượng lớn dữ liệu thông qua mạng nơ-ron nhân tạo để tìm hiểu về các tuỳ chọn của người dùng.

Ngoài ra, Instagram cũng sử dụng Deep Learning để tránh các hành vi bạo lực trên không gian mạng, chặn các bình luận vi phạm, không phù hợp,…

Hình 8: Mạng xã hội

2.1.4.4. Chăm sóc sức khoẻ :

Deep Learning cũng có đóng góp không nhỏ vào lĩnh vực y tế, trong đó phổ biến gồm có các mô hình dự đoán tình trạng bệnh, chẩn đoán ung thư, phân tích kết quả MRI, X-ray

Hình 9: Chăm sóc khách hàng

2.2. CÁC THƯ VIỆN – THUẬT TOÁN - MÔ HÌNH HỖ TRỢ :

Các thư viện hỗ trợ giải quyết vấn đề là **thư viện Scikit Learn và thư viện Keras**. Trong đó: Thư viện Scikit learn sẽ hỗ trợ về 2 mô hình máy học (Machine Learning) là **SVM** (Support vector machine) và **Random Forest** , Thư viện Keras sẽ hỗ trợ mô hình học sâu (Deep Learning) là **MLP**(Multi-Layer Perceptron)

2.2.1. Thư viện SCIKIT LEARN :

2.2.1.1. Giới thiệu thư viện :

Scikit learn (hay Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán machine learning và statistical modeling bao gồm***: clustering, classification, regression và dimensionality reduction***. Thư viện này cung cấp vô số thuật toán giúp xử lý các mô hình học máy hiệu quả và cung cấp các mô hình học máy như: ***SVM, decision tree, logistic regression, …***

2.2.1.2. Mô hình SVM (Support vector machine) :

SVM là viết tắt của cụm từ *support vector machine*. Đây là một thuật toán khá hiệu quả trong lớp các bài toán phân loại nhị phân và dự báo của học có giám sát. Thuật toán này có ưu điểm là hoạt động tốt đối với những mẫu dữ liệu có kích thước lớn và thường mang lại kết quả vượt trội so với lớp các thuật toán khác trong học có giám sát.

Mục tiêu của SVM là tìm ra một siêu phẳngtrong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành các phần tương ứng với lớp của chúng.

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC()
```

After being fitted, the model can t

```
>>> clf.predict([[2., 2.]])
array([1])
```

Hình 10: Ví dụ mô hình SVM

2.2.1.3. Mô hình Random Forest :

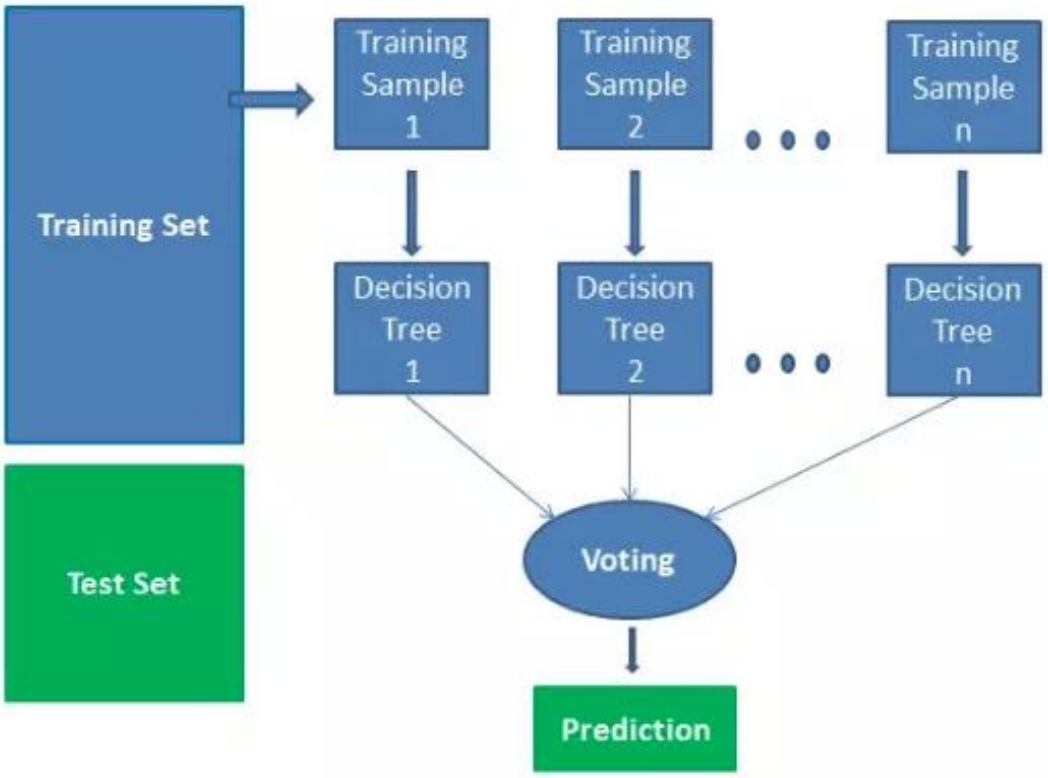
Random Forests là thuật toán học có giám sát (*supervised learning*). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Một khu rừng bao gồm cây cối. Người ta nói rằng càng có nhiều cây thì rừng càng mạnh.

Random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu. Nó cũng cung cấp một chỉ báo khá tốt về tầm quan trọng của tính năng. Random forests có nhiều ứng dụng, chẳng hạn như

công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các bệnh.

Cách hoạt động của mô hình Random Forest:

1. Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
2. Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
3. Hãy bỏ phiếu cho mỗi kết quả dự đoán.
4. Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Hình 11: Cách hoạt động của mô hình Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,
                           n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)

print(clf.predict([[0, 0, 0, 0]]))
```

Hình 12: Ví dụ Random Forest

2.2.1.3. Công cụ GridSearchCV :

GridSearchCV là một công cụ quan trọng trong Machine Learning được sử dụng để tìm kiếm siêu tham số tốt nhất cho một mô hình máy học cụ thể.

Cụ thể, GridSearchCV là một phương pháp tìm kiếm trên lưới siêu tham số khác nhau để tìm ra bộ siêu tham số tốt nhất cho mô hình. Siêu tham số là các tham số mà không được học từ dữ liệu mà được cài đặt trước, và chúng ảnh hưởng đến hiệu suất của mô hình.

Ví dụ: Trong mô hình Support Vector Machine (SVM), các siêu tham số như hằng số điều chỉnh lề (C) và hệ số kernel (gamma) có thể ảnh hưởng đến hiệu suất của mô hình.

```
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)
```

Hình 13: Ví dụ công cụ GridSearchCv cho mô hình SVC

Với GridSearchCV, ta định nghĩa các giá trị của các siêu tham số cần tìm kiếm, sau đó mô hình sẽ được đào tạo với tất cả các sự kết hợp của các giá trị siêu tham số này. Sau đó, mô hình được đánh giá bằng cách sử dụng một metric để đánh giá hiệu suất, ví dụ như độ chính xác. Cuối cùng, bộ siêu tham số tốt nhất được chọn dựa trên metric đánh giá hiệu suất tốt nhất.

2.2.2. Thư viện KERAS:

2.4.2.1. Giới thiệu thư viện :

Keras là một thư viện mã nguồn mở cho việc xây dựng các mô hình học sâu (*deep learning*) trên nền tảng Python. Nó cung cấp một cách đơn giản và dễ sử dụng để xây dựng các mô hình học sâu như **mạng nơ-ron** (*neural networks*), **mạng nơ-ron tích chập** (*convolutional neural networks*) và **mạng nơ-ron tái lập** (*recurrent neural networks*).

2.4.2.2. Mô hình MLP (Multi-Layer Perceptron):

Multi-Layer Perceptron (MLP) là một trong những kiến trúc mạng neuron cơ bản trong deep learning. Nó là một mạng neuron truyền thẳng (*feedforward neural network*) với nhiều lớp ẩn (*hidden layers*) giúp cho mô hình có khả năng học được các đặc trưng phức tạp của dữ liệu đầu vào.

Thư viện **Keras** cung cấp một số lớp (*layers*) để tạo nên kiến trúc MLP. Trong Keras, bạn có thể sử dụng lớp Dense để tạo các lớp ẩn trong mạng MLP. Lớp Dense cung cấp nhiều tính năng như khả năng thiết lập số lượng node (*unit*) trong lớp, hàm kích hoạt (*activation function*), hệ số trọng số (*weights*) và độ lệch (*bias*).

Để xây dựng một mô hình MLP trong Keras, bạn có thể sử dụng lớp **Sequential** để xếp các lớp ẩn lên nhau và kết thúc bằng một lớp đầu ra (output layer). Bạn cũng có thể sử dụng lớp Functional để tạo một mô hình đa nhánh hoặc tùy chỉnh hơn.

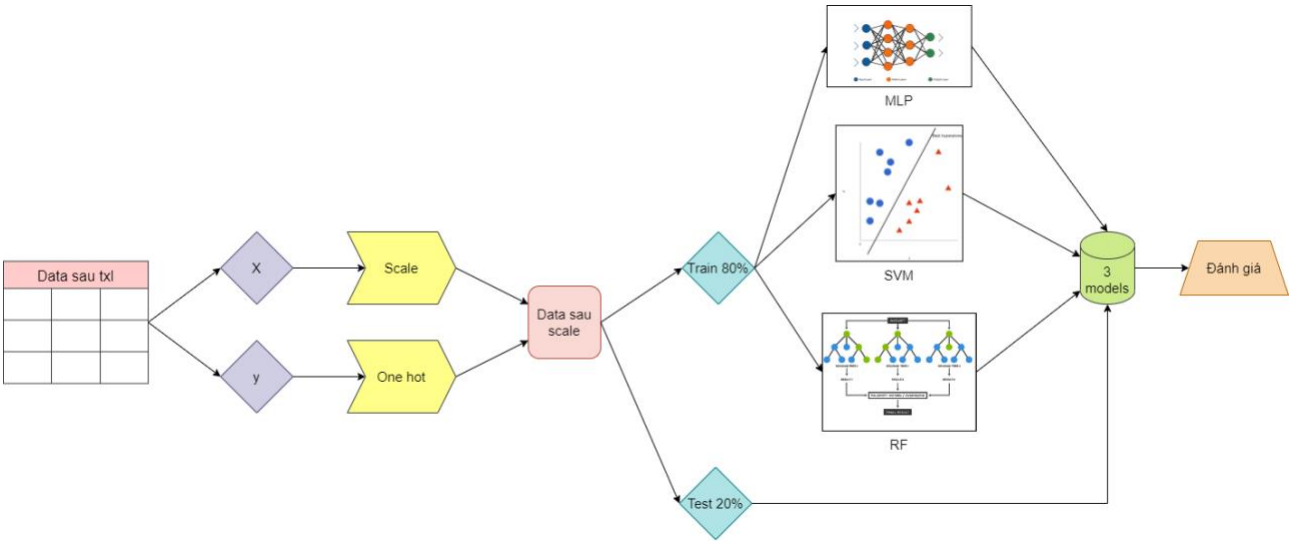
Sequential là một lớp mô hình được sử dụng để xây dựng một chuỗi các lớp mạng neuron liên tiếp. Mỗi lớp trong chuỗi có thể là một lớp hoàn toàn kết nối (*fully connected layer*), một lớp tích chập (*convolutional layer*), một lớp thảo luận (*recurrent layer*) hoặc bất kỳ loại lớp nào khác được hỗ trợ bởi Keras.

```
model = Sequential()
model.add(Dense(16, input_dim=20, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(4, activation='softmax'))
```

Hình 14: Ví dụ cho mô hình MLP

CHƯƠNG 3. MÔ HÌNH TRIỂN KHAI - THỰC HIỆN:

3.1. SƠ ĐỒ TỔNG QUÁT VỀ HƯỚNG THỰC HIỆN:



Hình 15: Sơ đồ tổng quát về phương hướng thực hiện

Giải thích sơ đồ phương hướng giải quyết bài toán phân tích:

- Sau khi tiến hành các phương pháp tiền xử lý ta sẽ có 1 tập dữ liệu, ta sẽ tách dữ liệu thành 2 tập: **tập X** là tập không chứa biến **Price Range** (*biến target*), **tập y** chỉ chứa biến **Price range**. Sau đó tiến hành train model.

+ Với tập X nhóm sẽ chuẩn hóa dữ liệu (scale) bằng phương pháp *Standard Scaler*.

+ Với tập y sẽ giữ nguyên đối với model máy học, đổi thành vector bằng phương pháp *Onehot-Encoding* đối với **model deep learning**.

- Sau khi hoàn thành các bước chuẩn hóa dữ liệu, nhóm sẽ tách dữ liệu thành 2 tập là train và test với tập train là 80% và tập test là 20% của dữ liệu rồi **tiến hành train 2 mô hình máy học là SVM và Random Forest, 1 mô hình deep learning là MLP.**

- Sau khi hoàn thành train mô hình sẽ tính các điểm số *Accuracy, MSE, F1-Score, Recall, Recision* để đánh giá mô hình nào tốt.

➔ Sau khi tìm được mô hình tốt để phân loại ta sẽ tìm được các đặc điểm tính năng của điện thoại để phân tích đánh giá.

3.2. TRIỂN KHAI - THỰC HIỆN:

3.2.1. Tiền xử lý dữ liệu :

- **Duplicates :** Kiểm tra dữ liệu bị trùng (duplicates)

```
duplicateRows = df[df.duplicated()]
print('Số dòng dữ liệu duplicates: ', len(duplicateRows))
```

```
Số dòng dữ liệu duplicates:  45
```

Hình 16: Số dòng dữ liệu bị trùng

Ta có thể thấy được bộ dữ liệu hiện có **45 dòng bị trùng lặp** → Tiến hành loại bỏ dữ liệu bị trùng, sau đó kiểm tra lại dữ liệu.

```
df = df.drop_duplicates()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2054 entries, 0 to 2098
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2054 non-null   int64  
 1   blue           2038 non-null   float64
 2   clock_speed    2054 non-null   float64
 3   dual_sim       2017 non-null   float64
 4   fc             2054 non-null   int64  
 5   four_g         2004 non-null   float64
 6   int_memory     2054 non-null   int64  
 7   m_dep          2054 non-null   float64
 8   mobile_wt      2054 non-null   int64  
 9   n_cores        2054 non-null   int64  
 10  pc             2001 non-null   float64
 11  px_height      2054 non-null   int64  
 12  px_width       2054 non-null   int64  
 13  ram            2054 non-null   int64  
 14  sc_h           1988 non-null   float64
 15  sc_w           2054 non-null   int64  
 16  talk_time      2054 non-null   int64  
 17  three_g        1947 non-null   float64
 18  touch_screen   2054 non-null   object 
 19  wifi           2054 non-null   object 
 20  price_range    2054 non-null   int64  
dtypes: float64(8), int64(11), object(2)
memory usage: 353.0+ KB
```

Hình 17: Loại bỏ các dòng bị trùng

❁ **Missing values :** Kiểm tra dữ liệu bị thiếu

```
#lấy ra những feature có missing value còn những feature k có thì sẽ lược bỏ đi để tránh gây nhiễu
df1 = df.dtypes   # Kiểu dữ liệu
a = pd.DataFrame(df1, columns = ['dtypes'])
df2 = len(df) - df.count()  # Số giá trị bị thiếu
b = pd.DataFrame(df2, columns = ['missing_val'])
df3 = (len(df) - df.count())/len(df)*100 # Tỷ lệ phần trăm giá trị bị thiếu
c = pd.DataFrame(df3, columns = ['per_missing'])

rlt=pd.concat([a, b, c], axis = 1)

# Sort the table by percentage of missing descending
mis_val_table_ren_columns = rlt[rlt.iloc[:,1] != 0].sort_values('per_missing', ascending=False).round(1)

print('Kết quả: \n', mis_val_table_ren_columns)
```

```
Kết quả:
           dtypes  missing_val  per_missing
three_g   float64          107          5.2
sc_h      float64           66          3.2
pc        float64           53          2.6
four_g    float64           50          2.4
dual_sim  float64           37          1.8
blue      float64           16          0.8
```

Hình 18: Số dữ liệu bị thiếu

- Bộ dữ liệu có **6 features** có missing values với số lượng lần lượt **là three_g (107), sc_h (66), pc (53), four_g (50), dual_sim (37) và blue (16)**

➔ Tiến hành thay thế dữ liệu bị thiếu :

+ Đối với dữ liệu dạng phân loại (**categorical**): Thay thế dữ liệu bị thiếu bằng giá trị xuất hiện nhiều nhất (**mode**)

```
# Điền dữ liệu Mode vào các Missing values
df['three_g'] = df['three_g'].fillna((df['three_g'].mode())[0])
df['four_g'] = df['four_g'].fillna((df['four_g'].mode())[0])
df['dual_sim'] = df['dual_sim'].fillna((df['dual_sim'].mode())[0])
df['blue'] = df['blue'].fillna((df['blue'].mode())[0])
```

Hình 19: Xử lý dữ liệu bị thiếu đối với dữ liệu dạng phân loại

+ Đối với dữ liệu dạng số (**numeric**): Thay thế dữ liệu bị thiếu bằng giá trị trung vị (**median**)

```
# Điền dữ liệu Median vào các Missing values
df['pc'] = df['pc'].fillna(df['pc'].median())
df['sc_h'] = df['sc_h'].fillna(df['sc_h'].median())
```

Hình 20: Xử lý dữ liệu bị thiếu đối với dữ liệu dạng số

- Sau đó ta sẽ kiểm tra lại dữ liệu sau khi xử lý missing value để kiểm tra xem tất cả giá trị thiếu đã được fill chưa.

```
df1 = df.dtypes   # Kiểu dữ liệu
a = pd.DataFrame(df1, columns = ['dtypes'])
df2 = len(df) - df.count()  # Số giá trị bị thiếu
b = pd.DataFrame(df2, columns = ['missing_val'])
df3 = (len(df) - df.count())/len(df)*100 # Tỷ lệ phần trăm giá trị bị thiếu
c = pd.DataFrame(df3, columns = ['per_missing'])
print('Kết quả: \n', pd.concat([a, b, c], axis = 1))
```

```
Kết quả:
                dtypes  missing_val  per_missing
battery_power    int64            0          0.0
blue           float64            0          0.0
clock_speed    float64            0          0.0
dual_sim       float64            0          0.0
fc               int64            0          0.0
four_g         float64            0          0.0
int_memory       int64            0          0.0
m_dep          float64            0          0.0
mobile_wt        int64            0          0.0
n_cores          int64            0          0.0
pc             float64            0          0.0
px_height        int64            0          0.0
px_width         int64            0          0.0
ram              int64            0          0.0
sc_h           float64            0          0.0
sc_w             int64            0          0.0
talk_time        int64            0          0.0
three_g        float64            0          0.0
touch_screen    object            0          0.0
wifi            object            0          0.0
price_range      int64            0          0.0
```

Hình 21: Số dữ liệu bị thiếu sau khi đã thay thế

```
len_df=df.shape[0]
print("Số dòng m mới = ",len_df )
print("Số cột n mới = ", df.shape[1])
```

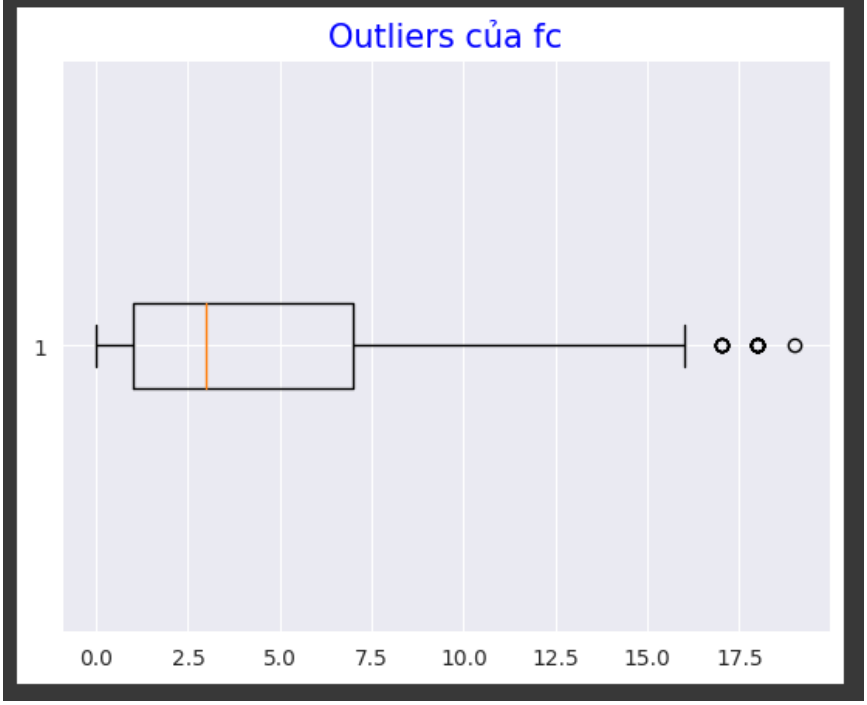
```
Số dòng m mới =  2054
Số cột n mới =  21
```

Hình 22: Dữ liệu sau khi đã thay thế dữ liệu bị thiếu

- **Outliers :** Kiểm tra dữ liệu bất thường
  Sau khi kiểm tra ta phát hiện ra có **2 features** có dữ liệu nhiễu : fc (front camera) và px_height (độ phân giải)

+ Đối với feature : fc (front camera)

```
plt.boxplot(df['fc'], vert=False)
plt.title('Outliers của fc', fontsize = 15, color = 'b')
plt.show()
```



Hình 23: Biểu đồ hộp của feature "fc"

Dựa vào biểu đồ hình hộp trên, thấy được rằng feature "fc" có sự xuất hiện outliers. Ta dựa vào quy tắc IQR để xác định số outlier của feature này.

```
Q1 = np.percentile(df['fc'] , 25)
Q3 = np.percentile(df['fc'] , 75)
IQR = Q3 - Q1
upper = Q3+1.5*IQR
lower = Q1-1.5*IQR
outliers_upper = df['fc'][(df['fc'] > upper)]
outliers_lower = df['fc'][(df['fc'] < lower)]
print('Số lượng outliers cận trên:',len(outliers_upper))
print('Số lượng outliers cận dưới:',len(outliers_lower))
```

```
Số lượng outliers cận trên: 18
Số lượng outliers cận dưới: 0
```

Hình 24: Số outliers của feature "fc"

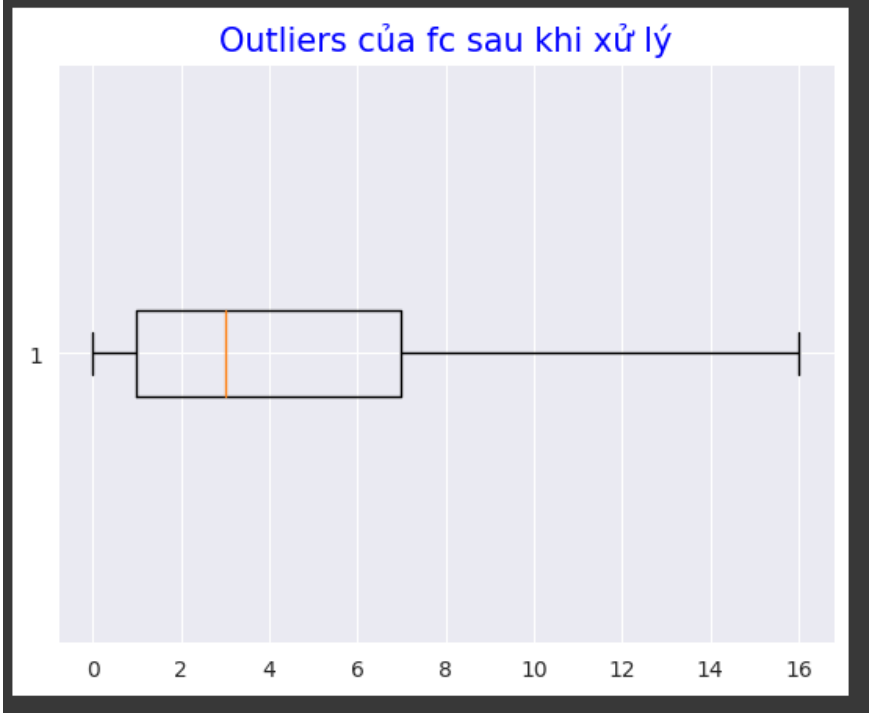
Dựa vào quy tắc trên ta tính được feature hiện có 18 outlier, ta sẽ tiến hành loại bỏ outlier cho feature này.

```
df = df[(df['fc'] >= lower) & (df['fc'] <= upper)]
```

Hình 25: Loại bỏ outliers của feature "fc"

Và đây là boxplot của feature "fc" sau khi loại bỏ outlier.

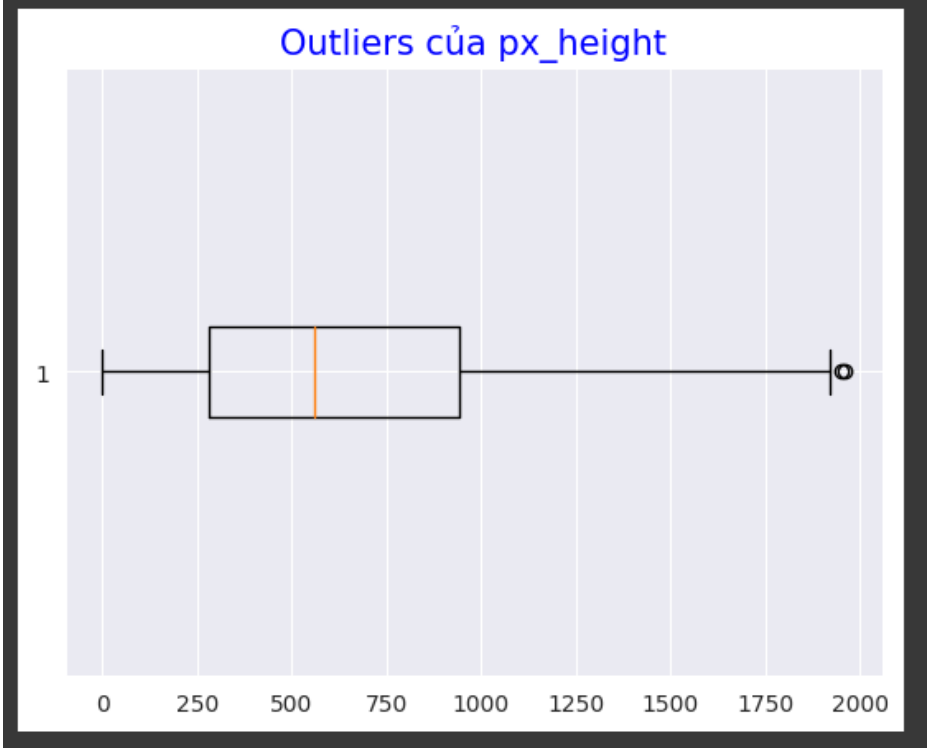
```
plt.boxplot(df['fc'], vert=False)
plt.title('Outliers của fc sau khi xử lý', fontsize = 15, color = 'b')
plt.show()
```



Hình 26: Kiểm tra lại outliers của feature "fc" sau khi đã loại bỏ

+ Đối với feature: px_height (Chiều cao độ phân giải)

```
plt.boxplot(df['px_height'], vert=False)
plt.title('Outliers của px_height', fontsize = 15, color = 'b')
plt.show()
```



Hình 27: Biểu đồ hộp của feature "px_height"

Dựa vào biểu đồ hình hộp trên, thấy được rằng feature "px_height" có sự xuất hiện outliers. Ta dựa vào quy tắc IQR để xác định số outlier của feature này.

```
Q1 = np.percentile(df['px_height'] , 25)
Q3 = np.percentile(df['px_height'] , 75)
IQR = Q3 - Q1
upper = Q3+1.5*IQR
lower = Q1-1.5*IQR
outliers_upper = df['px_height'][(df['px_height'] > upper)]
outliers_lower = df['px_height'][(df['px_height'] < lower)]
print('Số lượng outliers cận trên:',len(outliers_upper))
print('Số lượng outliers cận dưới:',len(outliers_lower))
```

```
Số lượng outliers cận trên: 2
Số lượng outliers cận dưới: 0
```

Hình 28: Số outliers của feature "px_height"

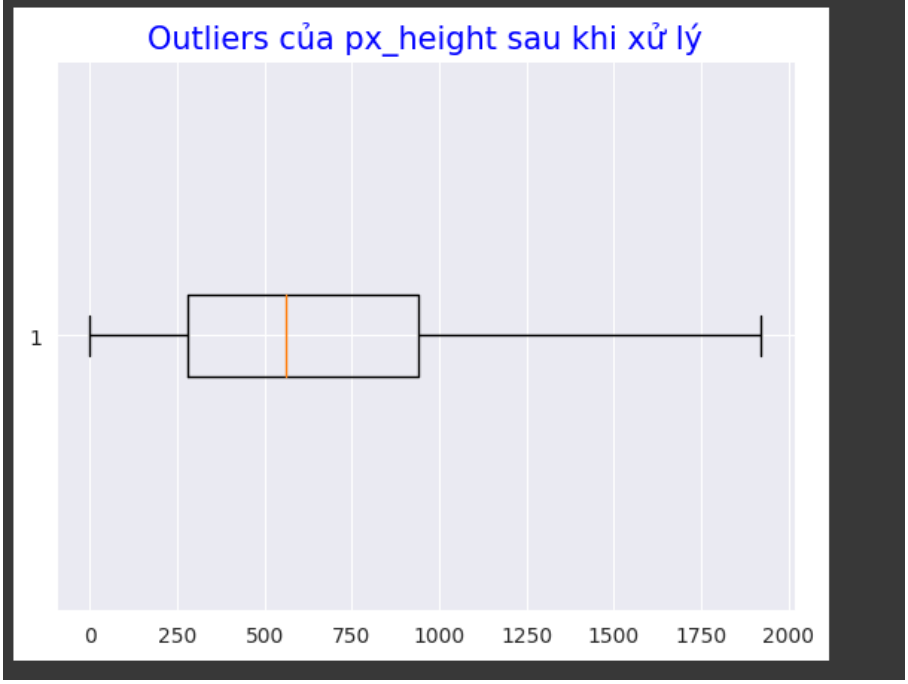
Dựa vào quy tắc trên ta tính được feature "px_height" hiện có 2 outlier, ta sẽ tiến hành loại bỏ outlier cho feature này.

```
df = df[(df['px_height'] >= lower) & (df['px_height'] <= upper)]
```

Hình 29: Loại bỏ outliers của feature "px_height"

Và đây là boxplot của feature "px_height" sau khi loại bỏ outlier.

```
plt.boxplot(df['px_height'], vert=False)
plt.title('Outliers của px_height sau khi xử lý', fontsize = 15, color = 'b')
plt.show()
```



Hình 30: Kiểm tra lại outliers của feature "px_height" sau khi đã loại bỏ

⇨ Để đảm bảo các outlier đã được loại bỏ, ta sẽ kiểm tra lại dữ liệu:

```
print("Số dòng m cũ  = ", len_df)
print("Số dòng m mới = ", df.shape[0])
print("Chênh lệch    =",len_df-df.shape[0])
print("Số cột n mới  = ", df.shape[1])
```

```
Số dòng m cũ  =  2054
Số dòng m mới =  2034
Chênh lệch    = 20
Số cột n mới  =  21
```

Hình 31: Dữ liệu sau khi loại bỏ outliers

➔ Sau khi loại bỏ outliers, dữ liệu **còn lại 2034 dòng**, ít hơn trước đó 20 dòng

- **Encoding**: Dữ liệu dạng phân loại

  - Khai báo **LabelEncoder**

```
encoder = LabelEncoder()
encoder.fit(["Yes", "No"])
```

```
▾ LabelEncoder
LabelEncoder()
```

Hình 32: Khai báo LabelEncoder

Thuật toán **LabelEncoder** sẽ thay đổi "**Yes**" thành **1** và "**No**" thành **0**

- Áp dụng **LabelEncoder** cho 2 biến dạng phân loại có giá trị "**Yes**" và "**No**" là 2 biến **"wifi" và " touch_screen".**

```
df['wifi'] = encoder.transform(df['wifi'])
df['touch_screen'] = encoder.transform(df['touch_screen'])
```

Hình 33: Áp dụng LabelEncoder

- Kiểm tra lại dữ liệu sau khi Encoder.

```
df.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	1.0	2.2	0.0	1	0.0	7	0.6	188	2	...	20	756	2549	9.0	7	19	0.0	0	1	1
1	1021	1.0	0.5	1.0	0	1.0	53	0.7	136	3	...	905	1988	2631	17.0	3	7	1.0	1	1	2
2	563	1.0	0.5	1.0	2	1.0	41	0.9	145	5	...	1263	1716	2603	11.0	2	9	1.0	1	1	2
3	615	1.0	2.5	0.0	0	1.0	10	0.8	131	6	...	1216	1786	2769	16.0	8	11	1.0	1	1	2
4	1821	1.0	1.2	0.0	13	1.0	44	0.6	141	2	...	1208	1212	1411	8.0	2	15	1.0	1	1	1

5 rows × 21 columns

Hình 34: Dữ liệu sau khi LabelEncoder

- **Tách dữ liệu:** Ta sẽ tách dữ liệu thành Target Y là "price_range" và Feature X là các features còn lại

```
X = df.drop(columns = ['price_range'])
y = df['price_range']
```

Hình 35: Tập Y là target và tập X

- **Chuẩn hóa dữ liệu (scale)**

  - Ta sử dụng thuật toán Standard Scaler để chuẩn hóa tập X

```
sc = StandardScaler()
X = sc.fit_transform(X)
print("Dữ liệu sau khi chuẩn hóa:")
X[0]
```

```
Dữ liệu sau khi chuẩn hóa:
array([-0.88935275,  0.63006192,  0.83088271, -1.60850912, -0.76395184,
       -1.60067849, -1.39179609,  0.33882734,  1.35246067, -1.1013441 ,
       -1.30326618, -1.41153316, -1.14136851,  0.40424596, -0.79084471,
        0.29217009,  1.46901622, -1.68455806, -2.11023295,  0.48319234])
```

Hình 36: Chuẩn hóa tập X với StandardScaler

❁ One hot-encoding :

- Sử dụng One hot-encoding để biến **tập Y** thành **vector** để huấn luyện deep learning MLP

```
ohe = OneHotEncoder()
y_MLP = ohe.fit_transform(y.to_numpy().reshape(-1, 1)).toarray()
print('One hot encoded:')
y_MLP[0:5]
```

```
One hot encoded:
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.]])
```

```
y[0:5]
```

```
0    1
1    2
2    2
3    2
4    1
```

Hình 37: One hot-encoding tập Y

❁ Tách bộ dữ liệu thành 2 tập train và test để huấn luyện mô hình:

- Ta sử dụng train_test_split để tách dữ liệu thành 2 tập train và test với tỉ lệ test 20% train 80%

```
random_state = 11

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

y_MLP_train, y_MLP_test = train_test_split(y_MLP, test_size = 0.2, random_state = random_state)
```

Hình 38: Tách dữ liệu thành 2 tập train và test

3.3.2. Huấn luyện mô hình (Train model):

3.3.2.1. Mô hình Máy học (Machine Learning):

Lựa chọn mô hình và các chỉ số:

- Hai mô hình máy học được lựa chọn là *SVM (Support Vector Machine) và Random Forest*

```
classifier = [SVC(random_state = random_state, probability = True),
              RandomForestClassifier(random_state = random_state)]
```

Hình 39: Lựa chọn mô hình máy học

- Khai báo các chỉ số của 2 mô hình

```
svc_param_grid = {"kernel" : ["rbf"],
                  "gamma": [0.001, 0.01, 0.1, 1],
                  "C": [0.1, 1,10,50,100,200,300,1000]}

rf_param_grid = {"max_features": [1,3,10],
                 "min_samples_split":[2,3,10],
                 "min_samples_leaf":[1,3,10],
                 "bootstrap":[False],
                 "n_estimators":[100,300],
                 "criterion":["gini"]}
```

Hình 40: Các chỉ số của mô hình

Trong đó:

- *Đối với mô hình SVM*:
  - **gamma:** hệ số karnel, mặc định là 1/n (features)
  - **C:** hệ số chính quy, mặc định là 1
- Đối với mô hình Random Forest:
  - **n_estimators:** số lượng tree trong forest, mặc định là 100
  - **min_samples_split:** số lượng mẫu tối thiểu để tách nút trong cây, mặc định là 2
  - **min_samples_leaf:** số lượng mẫu tối thiểu cần có trong 1 lá, mặc định là 1
  - **max_features:** số lượng features cần thiết để xét sự phân chia

⇨ Vì chưa biết chỉ số nào tốt nhất cho mô hình nên nhóm đã khai báo tất cả các chỉ số để lựa chọn

Huấn luyện mô hình :

- Sử dụng **GridSearchCV** để huấn luyện mô hình bằng tất cả các trường hợp theo chỉ số mô hình đã khai báo.
- Chọn ra chỉ số tốt nhất cho mô hình theo độ chính xác (accuracy) cao nhất.

```
for i in range(len(classifier)):
    print("------------------------------------------------------------------------------")
    clf = GridSearchCV(classifier[i],
                       param_grid=classifier_param[i],
                       cv = StratifiedKFold(n_splits = 10),
                       scoring = "accuracy",
                       n_jobs = -1,verbose = 2)

    clf.fit(X_train,y_train)

    cv_result.append(clf.best_score_)

    mean_squared_errors.append(mean_squared_error(y_test,clf.predict(X_test)))

    roc_auc_scores.append(roc_auc_score(y_test, clf.predict_proba(X_test), multi_class='ovr'))

    recall_scores.append(recall_score(y_test, clf.predict(X_test), average='weighted'))

    precision_scores.append(precision_score(y_test, clf.predict(X_test), average='weighted'))

    f1_scores.append(f1_score(y_test, clf.predict(X_test), average='weighted'))

    best_estimators.append(clf.best_estimator_)
```

Hình 41: Huấn luyện mô hình và chọn ra chỉ số tốt nhất cho mô hình

```
        print("Model: {}".format(classifier[i]))
        print("Accuracy: %{}".format(round(cv_result[i]*100,2)))
        print("MSE: {}".format(mean_squared_errors[i]))
        print("ROC AUC: {}".format(roc_auc_scores[i]))
        print("Recall: {}".format(recall_scores[i]))
        print("Precision: {}".format(precision_scores[i]))
        print("F1-Score: {}".format(f1_scores[i]))
        print("Best Estimator: {}".format(clf.best_estimator_))

    print("------------------------------------------------------------------------------")

    sns.set_style("darkgrid")
    cv_results = pd.DataFrame({"Accuracy":cv_result,
                               "MSE":mean_squared_errors,
                               "ROC AUC":roc_auc_scores,
                               "Recall": recall_scores,
                               "Precision": precision_scores,
                               "F1-Score":f1_scores,
                               "Models":[ "SVC",
                                          "RandomForestClassifier"]})

    cv_results.index = cv_results["Models"]

    cv_results  = cv_results.drop(["Models"], axis = 1)

    f,ax = plt.subplots(figsize=(14,10))

    sns.heatmap(cv_results, annot=True,cmap = "Blues",fmt= '.3f',
                ax=ax,linewidths = 5, cbar = False,
                annot_kws={"size": 18})

    plt.xticks(size = 18)
    plt.yticks(size = 18, rotation = 0)
    plt.ylabel("Models")
    plt.title("Grid Search Results", size = 16)
    plt.show()
```

Hình 42: Biểu diễn kết quả huấn luyện và các điểm số đánh giá mô hình

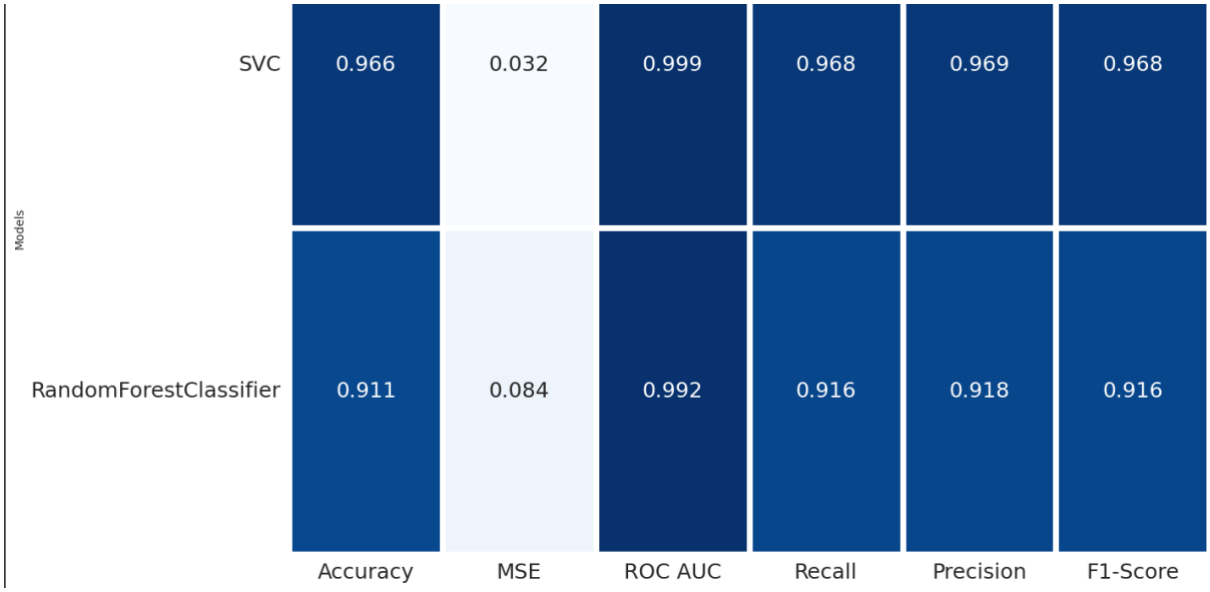
```
Fitting 10 folds for each of 32 candidates, totalling 320 fits
Model: SVC(probability=True, random_state=11)
Accuracy: %96.62
MSE: 0.03194103194103194
ROC AUC: 0.9993189789030328
Recall: 0.9680589680589681
Precision: 0.9690405785834477
F1-Score: 0.9681889964698176
Best Estimator: SVC(C=1000, gamma=0.001, probability=True, random_state=11)
--------------------------------------------------------------------------------
Fitting 10 folds for each of 54 candidates, totalling 540 fits
Model: RandomForestClassifier(random_state=11)
Accuracy: %91.15
MSE: 0.08353808353808354
ROC AUC: 0.9920733411950634
Recall: 0.9164619164619164
Precision: 0.9176432182451647
F1-Score: 0.9163883928971782
Best Estimator: RandomForestClassifier(bootstrap=False, max_features=10, min_samples_leaf=3,
                       n_estimators=300, random_state=11)
```

Hình 43: Kết quả chỉ số tốt nhất cho mô hình và điểm số đánh giá

Sau khi huấn luyện mô hình với thuật toán GridSearchCV nhận được kết quả như sau:

+ Đối với **mô hình SVM** qua 32 trường hợp mô hình ta có được chỉ số tốt nhất cho mô hình này là **c = 1000, gamma = 0.001.**

+ Đối với **mô hình Random Forrest**: qua 54 trường hợp ta có được chỉ số tốt nhất cho mô hình này là **max_freatures = 10, min_samples_leaf = 3, n_estimators = 300.**



Hình 44: Điểm số đánh giá của mô hình

⇨ Dựa vào điểm số chính xác (**accuracy**) cao (96,6% và 91,1%) và Mean Squared Erorr (**MSE**) thấp (0.032 và 0.084) cho thấy mô hình SVM và Random Forest đều rất phù hợp cho bộ dữ liệu → Mô hình tốt hơn là SVM với accuracy là 96,6%

3.3.2.1. Mô hình Học sâu (Deep Learning):

- Khai báo model và các node cho mạng lưới neural :

```
model = Sequential()
model.add(Dense(16, input_dim=20, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(4, activation='softmax'))
```

Hình 45: Model Sequential và các node

- **Node đầu tiên** có số chiều input (input_dim) là 20 và số chiều output là 16, activation function là "relu" giúp tăng tốc quá trình training
- **Node 2** có số chiều input là 16 từ node 1 và số chiều output là 12, activation function cũng là "relu"
- **Node cuối cùng** có số chiều input là 12 từ node 2 và số chiều output là 4 theo 4 class, vì là node cuối nên activation function là "softmax" để tính đầu ra xác suất dự đoán rơi vào các class.

- Khai báo các chỉ số của model :

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_MLP_train, epochs=120, batch_size=64)
```

Hình 46: Các chỉ số của model

- **Batch_size :** số lượng dữ liệu sử dụng trong 1 lần cập nhập tham số. Batch_size thường dùng là 2^n để tính toán nhanh và ở dây dữ liệu không nhiều (hơn 2000 dòng) nên dùng batch_size là 64 hoặc 32
- **Eponchs:** số lần duyệt qua hết các dữ liệu trong tập huấn luyện.

```
Epoch 112/120
26/26 [==============================] - 1s 23ms/step - loss: 0.0315 - accuracy: 0.9982
Epoch 113/120
26/26 [==============================] - 1s 22ms/step - loss: 0.0318 - accuracy: 0.9969
Epoch 114/120
26/26 [==============================] - 1s 21ms/step - loss: 0.0308 - accuracy: 0.9975
Epoch 115/120
26/26 [==============================] - 1s 22ms/step - loss: 0.0312 - accuracy: 0.9963
Epoch 116/120
26/26 [==============================] - 1s 22ms/step - loss: 0.0301 - accuracy: 0.9982
Epoch 117/120
26/26 [==============================] - 1s 22ms/step - loss: 0.0297 - accuracy: 0.9975
Epoch 118/120
26/26 [==============================] - 1s 21ms/step - loss: 0.0290 - accuracy: 0.9982
Epoch 119/120
26/26 [==============================] - 1s 22ms/step - loss: 0.0289 - accuracy: 0.9982
Epoch 120/120
26/26 [==============================] - 1s 24ms/step - loss: 0.0283 - accuracy: 0.9975
```

Hình 47: Kết quả huấn luyện

- Dự báo và đổi kết quả y từ vector về thành số như cũ để tính các điểm số:

```
y_pred = model.predict(X_test)

pred = list()
for i in range(len(y_pred)):
    pred.append(np.argmax(y_pred[i]))
```

```
13/13 [==============================] - 0s 2ms/step
```

Hình 48: Dự báo và đổi y từ vector thành số

- Tính các điểm số đánh giá mô hình:

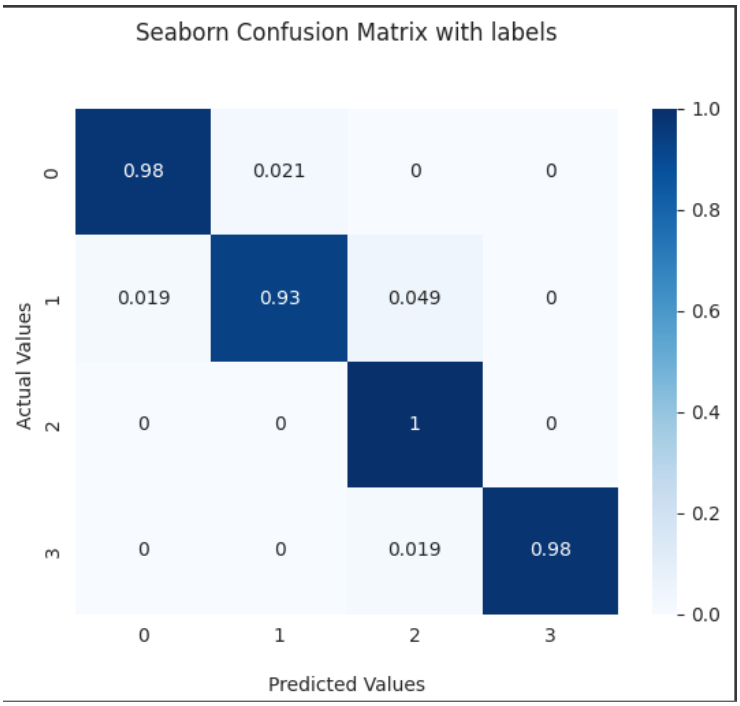
```
print("Accuracy: %{}".format(round(accuracy_score(pred, y_test)*100,2)))
print("MSE: {}".format(mean_squared_error(pred, y_test)))
print("Recall: {}".format(recall_score(pred, y_test, average='weighted')))
print("Precision: {}".format(precision_score(pred, y_test, average='weighted')))
print("F1-Score: {}".format(f1_score(pred, y_test, average='weighted')))
```

```
Accuracy: %97.3
MSE: 0.02702702702702703
Recall: 0.972972972972973
Precision: 0.9739032942916437
F1-Score: 0.9729946953176158
```

Hình 49: Các điểm số của mô hình deep learning

⇨ Sau khi huấn luyện mô hình deep learning MLP cho được kết quả với điểm số chính xác (**accuracy**) rất cao là 97,3% và Mean Squared Erorr (**MSE**) thấp là 0.027 cho thấy mô hình huấn luyện **rất tốt**.

- Vẽ ma trận nhầm lẫn để đánh giá mô hình:

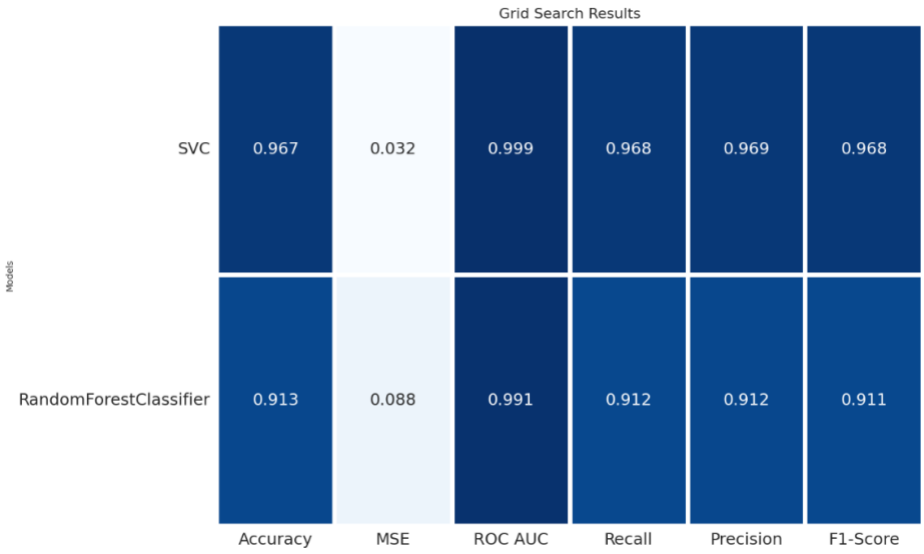


Hình 50: Ma trận nhầm lẫn

- Ma trận nhầm lẫn gồm các hàng là dữ liệu thực tế, các cột là dữ liệu dự báo. Các giá trị sẽ là tỉ lệ dự báo của dữ liệu thực tế theo dữ liệu dự báo
- Thông qua ma trận nhầm lẫn cho thấy các lớp 0, 2, 3 được dự báo chính xác (98%, 100%, 98%), riêng lớp 1 dự báo chưa được chính xác bằng với 93%.

3.3. PHÂN TÍCH – ĐÁNH GIÁ:

3.3.1. Lựa chọn mô hình tốt nhất :



Hình 51: Các chỉ số của mô hình SVC và Random Forest

```
Accuracy: %97.3
MSE: 0.02702702702702703
Recall: 0.972972972972973
Precision: 0.9739032942916437
F1-Score: 0.9729946953176158
```

Hình 52: Các chỉ số của mô hình học sâu MLP

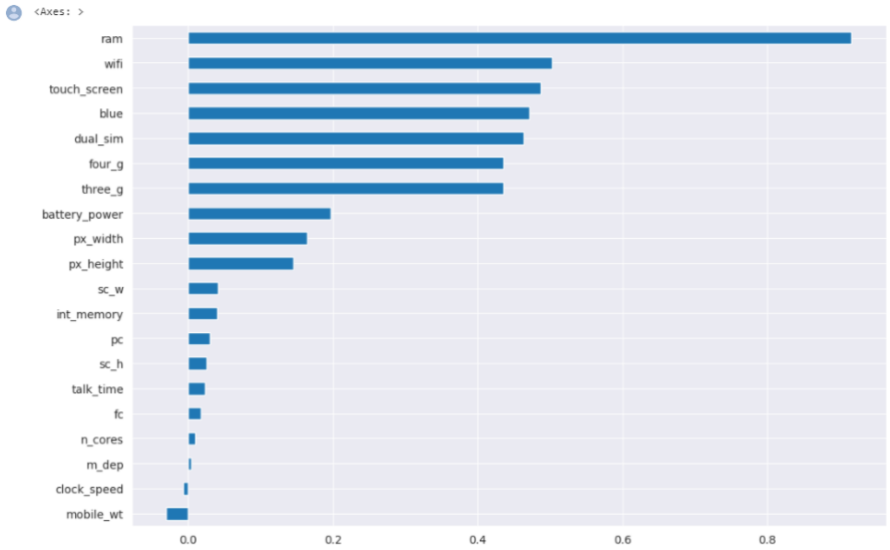
Thông qua 3 mô hình trên ta có thể thấy được rằng:
+ Giữa với 2 mô hình máy học với nhau thì có thể thấy được mô hình SVC có các chỉ số tốt hơn Random Forest (Accuracy: 96,7%>91,3%, MSE: 0,32 < 0,88,...)
+ Nhưng khi xét cả mô hình học sâu MLP vào ta có thể thấy được mô hình học sâu có các chỉ số tốt hơn cả 2 mô hình máy học trên (Accuracy: cao tận 97,3%, và tỉ lệ MSE: 0.027 ,... → Đây là các chỉ số rất tốt)

⇨ **Quyết định chọn mô hình học sâu MLP là mô hình tốt nhất để phân loại phân hạng giá điện thoại.**

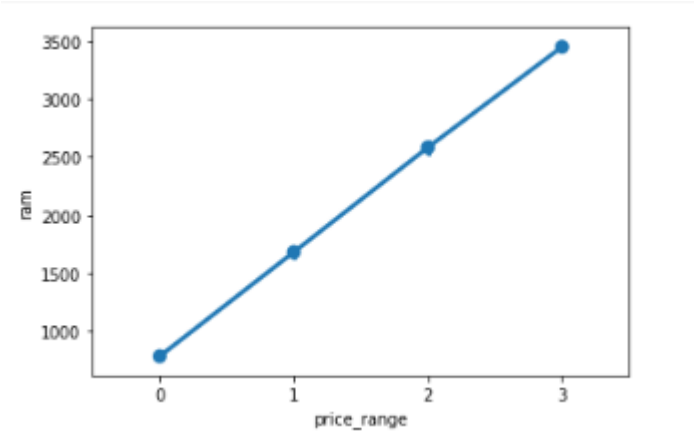
3.3.2. Phân tích sự ảnh hưởng của các biến đến giá – “Price Range”

- Các tín năng (Biến) ảnh hưởng mạnh đến giá – “Price_Range” :

Hình 53: HeatMap thể hiện tương quan giữa các biến



Hình 54: : Biểu đồ thể hiện mức độ ảnh hưởng giữa biến price_range và các biến còn lại.



Hình 55: Biểu đồ thể hiện mối tương quan giữa biến price_range và biến ram

Nhận xét:

Thông qua các biểu đồ trên ta thấy được rằng :

+ Biến "ram" chính là biến có ảnh hưởng mạnh nhất đến biến price_range (khoảng 0.92).

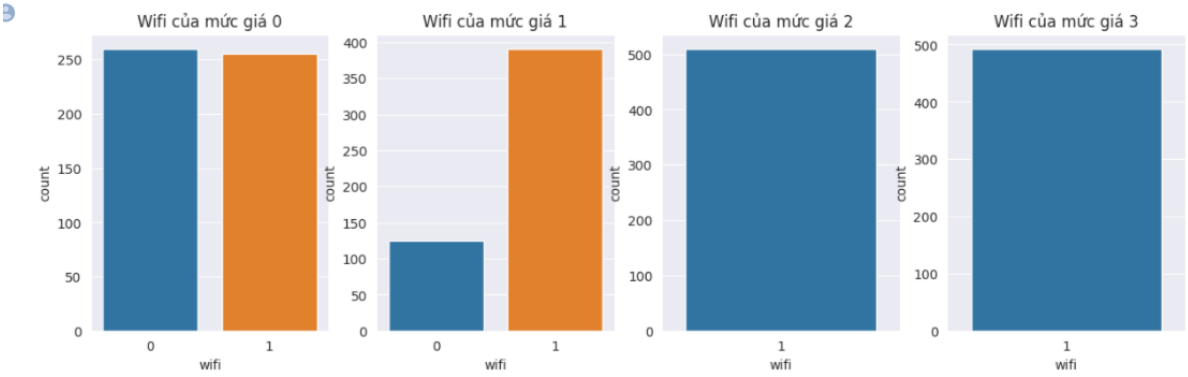
+ "Ram" càng lớn thì mức giá của nó cũng cao theo.

Giải thích:

- Tại vì "RAM" là một thành phần rất quan trọng trong điện thoại di động và các thiết bị điện tử khác. Nó được sử dụng để lưu trữ các ứng dụng và dữ liệu tạm thời khi chúng đang được sử dụng. Khi một điện thoại có dung lượng RAM thấp, nó sẽ không thể xử lý các ứng dụng và tác vụ nhanh chóng và hiệu quả như một điện thoại có dung lượng RAM cao hơn.

➔ Vì vậy, khi sản xuất một điện thoại với dung lượng RAM cao hơn, các nhà sản xuất phải bỏ ra nhiều chi phí hơn để mua các chip RAM đắt tiền hơn và tích hợp chúng vào thiết kế điện thoại. Những chi phí này cuối cùng sẽ được truyền cho người tiêu dùng thông qua giá bán của điện thoại. Vì vậy, điện thoại có dung lượng RAM cao hơn thường có giá bán cao hơn so với các điện thoại có dung lượng RAM thấp hơn. Nên "RAM" ảnh hưởng mạnh đến "Price_Range"

- Các mức giá theo từng biến phân loại – Category:



Hình 56: Tương quan giữa biến Wifi với từng mức giá (Price_Range)

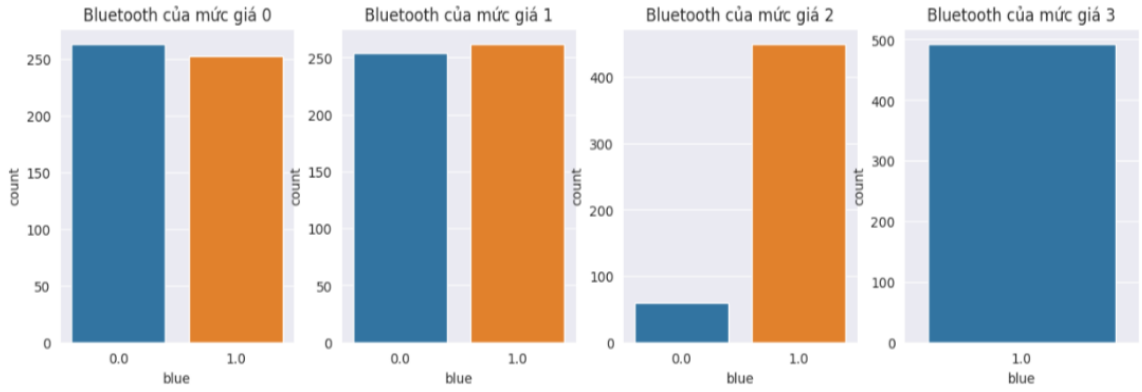
Nhận xét:

+ Ở mức giá 0 (thấp) thì số điện thoại có thể kết nối wifi và không kết nối wifi khi được bán ra gần như xấp xỉ bằng nhau.

+ Khi qua mức giá 1 (trung bình) thì số lượng điện thoại không có wifi lại thấp dần hơn so với điện thoại có wifi, và lên dần đến mức 2 (mức giá cao), mức 3(rất cao) tất số điện thoại được bán ra đều có khả năng kết nối wifi

Giải thích:

Với sự phát triển của công nghệ và nhu cầu ngày càng cao của người dùng, việc tích hợp tính năng kết nối wifi vào các mẫu điện thoại đã trở thành điều rất phổ biến và thông thường. ➔ Vậy nên với mức giá càng cao càng phải có khả năng kết nối wifi để phục vụ cho nhu cầu người dùng.



Hình 57: Tương quan giữa biến Blue với từng mức giá (Price_Range)

Nhận xét:

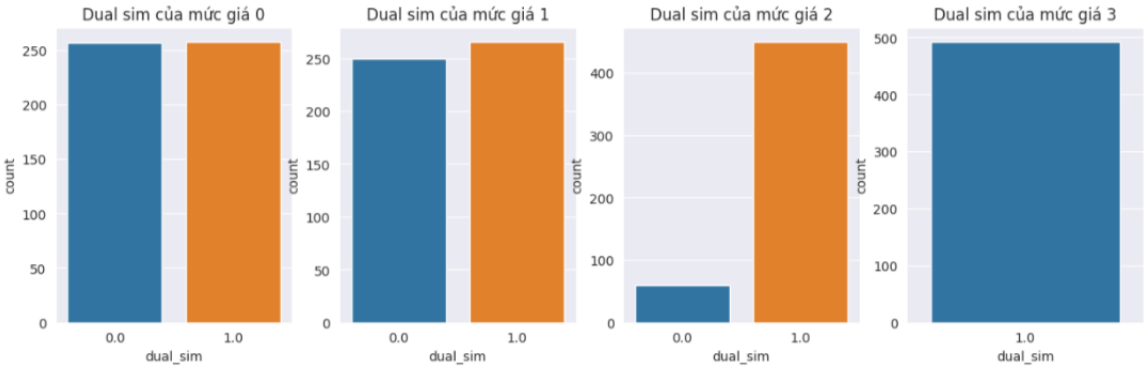
+ Ở mức giá 0 (thấp) thì số lượng điện thoại có Bluetooth và không có cũng xấp xỉ bằng nhau đồng thời qua mức giá 1 (trung bình) thì tỉ lệ cũng như vậy.

+ Qua mức giá 2 (cao) vẫn có điện thoại không có Bluetooth được bán nhưng chiếm số lượng ít cho thấy việc có Bluetooth hay không thì vẫn được mua bình thường dù nó có mức giá cao.

+ Qua mức giá 3 (rất cao) thì chắc chắn là phải có đầy đủ mọi chức năng thì mới nằm ở mức giá như vậy nên không cần phải bàn gì đến nó.

Giải thích:

Bluetooth là một công nghệ kết nối không dây giữa các thiết bị điện tử, và được tích hợp trong các smartphone và điện thoại di động thông thường. Vậy nên việc mức giá thấp và trung bình không có bluetooth là việc bình thường, tại có thể ở mức này việc kết nối thông qua Bluetooth của nó không cần thiết. Còn mức giá cao và rất cao phải có để tương hợp với giá của điện thoại.



Hình 58: Tương quan giữa biến Dual_sim với từng mức giá (Price_Range)

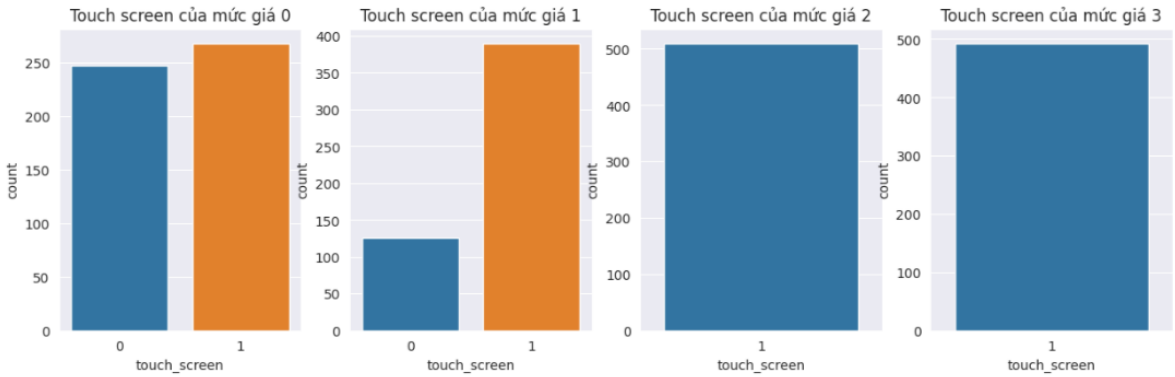
Nhận xét:

+ Ở mức giá 0 (thấp), mức 1(trung bình) thì số lượng điện thoại có hỗ trợ sim kép và không có cũng xấp xỉ bằng nhau.

+ Qua mức giá 2 (cao) số lượng điện thoại có hỗ trợ sim kép nhiều hơn gấp mấy lần so với không hỗ trợ.

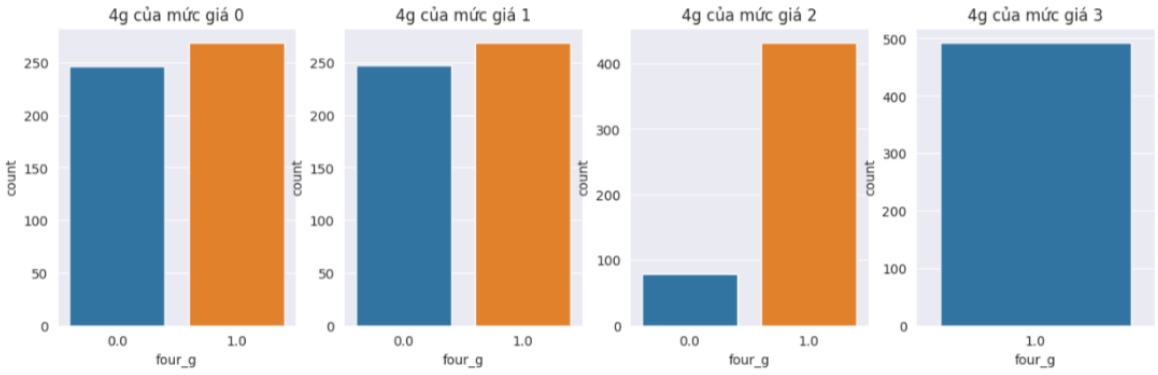
+ Qua mức giá 3 (rất cao) thì chắc chắn là phải có đầy đủ mọi chức năng thì mới nằm ở mức giá như vậy nên không cần phải bàn gì đến nó.

➔ Việc sử dụng 2 hay 1 sim thì cũng không quan trọng, chủ yếu là do nhu cầu của mọi người. Hai sim tuy có thể mang đến nhiều chức năng hơn là 1 sim nhưng nó lại gây hao pin hơn, và gây nhầm lẫn khi mỗi lần muốn sử dụng một sim nào đó.

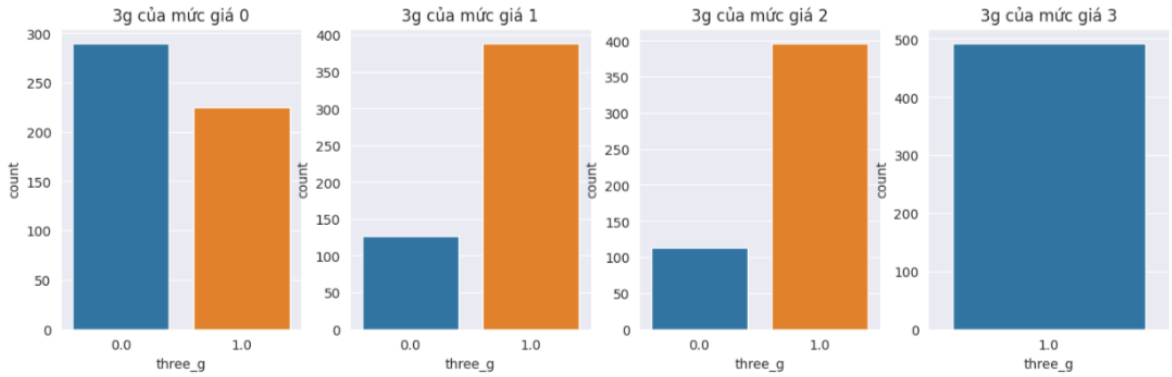


Hình 59: Tương quan giữa biến Touch_screen với từng mức giá (Price_Range)

Nhận xét: Với công nghệ ngày nay phát triển việc mua một chiếc điện thoại cảm ứng không còn quá khó khăn so với lúc trước nữa. Đa phần điện thoại ngày nay đều có cảm ứng cả thậm chí đối với những chiếc điện thoại ở mức 0 hay 1.



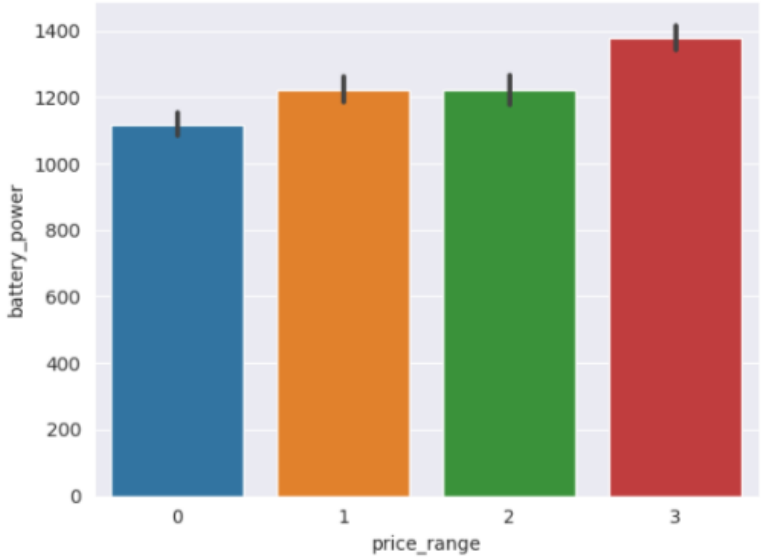
Hình 60: Tương quan giữa biến 4G với từng mức giá (Price_Range)



Hình 61: Tương quan giữa biến 3G với từng mức giá (Price_Range)

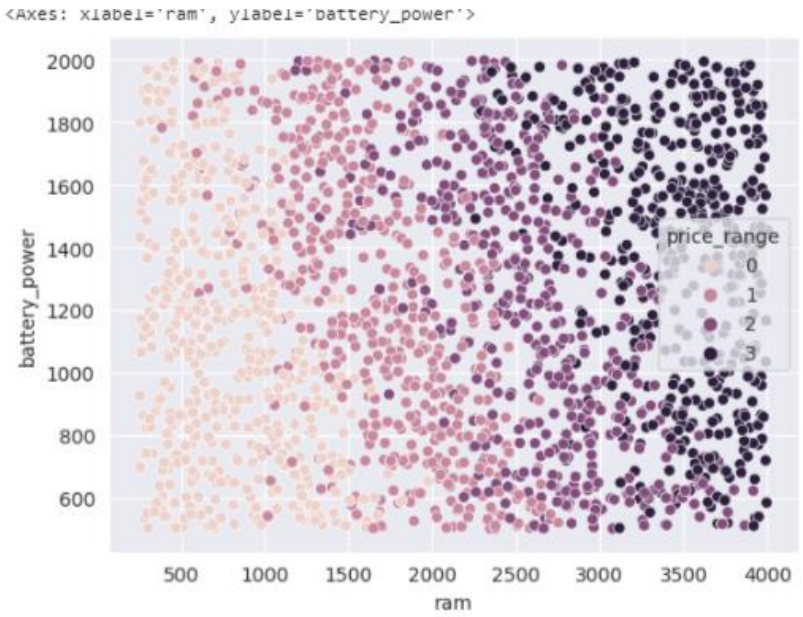
Nhận xét: Đối với 3G và 4G thì ở hai mức giá 2 và 3 thì gần như là giống nhau chỉ khác biệt ở 2 mức giá 0 và 1. Ở mức giá 0 thì lượng mua điện thoại không có 3G cao hơn so với 4G còn ở mức giá 1 thì ngược lại cho thấy được tại thời điểm 2018 4G vẫn chưa phát triển mạnh đa số mọi người biết đến 3G nhiều hơn nên lượng mua cũng cao hơn so với những chiếc điện thoại 4G.

- Các mức giá theo từng biến giá trị :



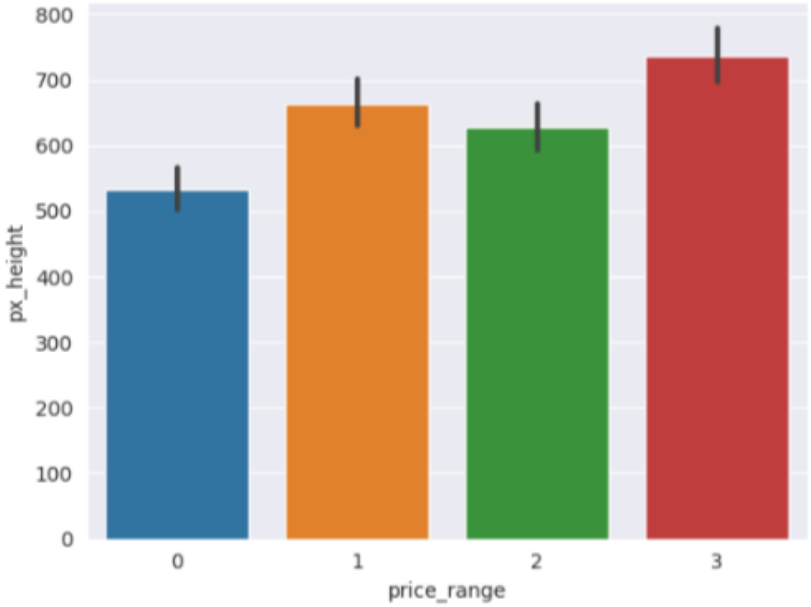
Hình 62: Biểu đồ thể hiện mối tương quan giữa biến battery_power với biến price_range

Nhận xét: Cũng giống như "Ram" thì biến "battery_power" cũng có mối quan hệ tỷ lệ thuận so với biến price_range lượng pin càng cao thì giá cũng cao theo. Tuy nhiên khi nhìn chung thì lượng pin của 4 mức giá đều rất cao (Đều trên 1000) và cách biệt không quá lớn giữa các mức giá với nhau → Lượng pin không ảnh hưởng đến mức giá.

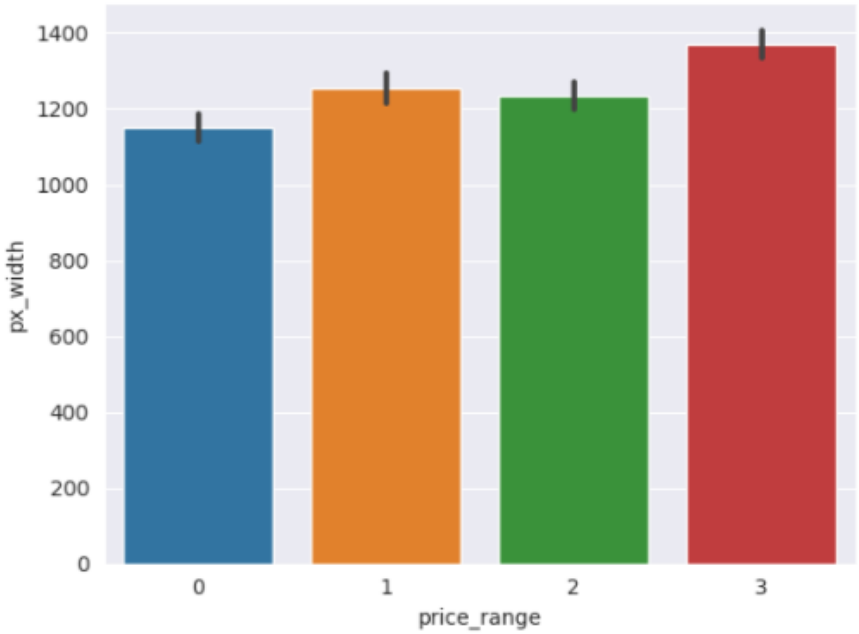


Hình 63: Biểu đồ thể hiện mối tương quan giữa 2 biến battery_power và biến ram với biến price_range

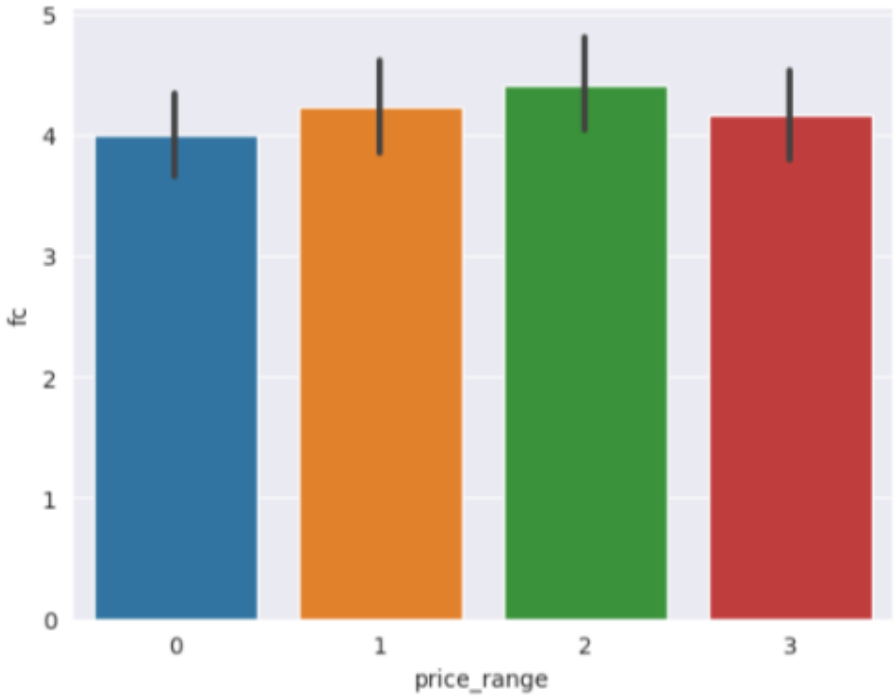
Nhận xét : Thông qua biểu đồ Cluster trên ta thấy được rằng 4 mức giá được phân ra khá rõ rệt khi xét theo hướng biến "Ram" → Ram càng cao thì các điểm mức giá càng rõ. Nhưng đối với biến "Battery_power" dù ở lượng pin nào thì mức giá đều trải đều từ 0 → 3 ➔ Mức giá ảnh hưởng mạnh bởi biến "Ram" và không bị ảnh hưởng nhiều bởi biến "Battery_power"



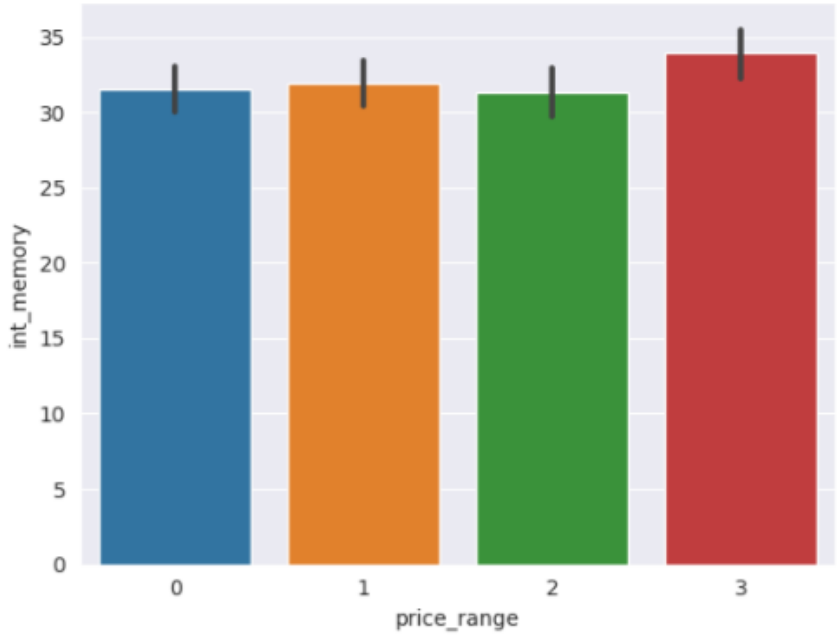
Hình 64: Biểu đồ thể hiện mối tương quan giữa biến pc_height với biến price_range



Hình 65: Biểu đồ thể hiện mối tương quan giữa biến pc_width với biến price_range



Hình 66: Biểu đồ thể hiện mối tương quan giữa biến fc với biến price_range



Hình 67: Biểu đồ thể hiện mối tương quan giữa biến int_memory với biến price_range

Nhận xét : Qua những biểu đồ trên (64→ 67) ta thấy hầu như đều có các giá trị ngang nhau xấp xỉ nhau ở từng mức giá điều đó cho thấy những những thông số trên dù ở mức giá nào cũng có thể đáp ứng được các tính năng đó. Ngoài ra, những thông số trên là những thông số về ngoại hình điện thoại, do đó dù mức giá thấp hay cao họ cũng sẽ đẩy mạnh về các chỉ số này.

CHƯƠNG 4. TỔNG KẾT

4.1. KẾT LUẬN:

Qua những phân tích từ bộ dữ liệu trên ta kết luận được rằng "**RAM**" là yếu tố quan trọng ảnh hưởng mạnh mẽ đến mức giá của sản phẩm. Vì khi sản xuất một điện thoại với dung lượng RAM cao hơn, các nhà sản xuất phải bỏ ra nhiều chi phí hơn để mua các chip RAM đắt tiền hơn và tích hợp chúng vào thiết kế điện thoại. Điều này dẫn đến việc RAM càng cao thì mức giá sẽ càng tăng lên.

Ngoài ra "RAM", còn có những tính năng khác như hỗ trợ kết nối wifi, bluetooth, sim kép, kết nối 3G/4G, khả năng cảm ứng cũng góp phần ảnh hưởng đến mức giá của điện thoại cụ thể là :

- Ở mức giá 0 (thấp): Hai cột ở các biểu đồ ở mức giá này đều xấp xỉ nhau điều đó chứng tỏ đa phần ở những chiếc máy được bán ra ở phân khúc giá này là những chiếc điện thoại bấm thông thường và những chiếc điện thoại thông minh.
- Ở mức giá 1 (trung bình): Đây là mức giá vừa đủ để sở hữu chiếc điện thoại thông minh hỗ trợ các tín năng về cảm ứng, wifi, mạng 3G. Vậy nên đối với khách hàng, việc bỏ ra số tiền với mức giá trung bình sẽ mong muốn nhận lại một điện thoại thông minh hơn là một chiếc điện thoại bấm bình thường.
- Ở mức giá 2 (cao): ta thấy ở phân khúc này thì không còn xuất hiện những chiếc điện thoại bấm được bán ra nữa thể hiện thông qua biểu đồ 2 biến touch_screen và wifi. Cũng dễ hiểu vì đã là 1 chiếc điện thoại ở phân khúc giá cao rồi thì người tiêu dùng sẽ mong muốn một chiếc điện thoại thông minh hoàn toàn hơn là 1 chiếc điện thoại bấm.
- Ở múc giá 3 (rất cao): Ở phân khúc này thì lựa chọn 1 chiếc điện thoại thông minh đầy đủ chức năng tác vụ là 1 lựa chọn không thể nào hơn.

Mặc khác, có các yếu tố về ngoại hình điện thoại, lượng Pin của tất cả điện thoại ở tất cả mức giá đều giống nhau. Tức là các chỉ số về ngoại hình, lượng Pin sử dụng của điện thoại không ảnh hưởng đến mức giá - "Price_range". Vậy nên ở những mức giá nào thì sản phẩm cũng đáp ứng đủ các yếu tố về ngoại hình, lượng Pin phù hợp phục vụ đủ cho nhu cầu khách hàng.

4.2. HƯỚNG PHÁT TRIỂN:

Tóm lại, bộ dữ liệu **Mobile Price Classification** là một bộ dữ liệu rất hữu ích để phát triển các mô hình học máy trong lĩnh vực phân loại giá cả của điện thoại di động. Kết quả của nghiên cứu này có thể được áp dụng cho các ứng dụng như các trang thương mại điện tử, các trang web đánh giá sản phẩm hay các hệ thống hỗ trợ quyết định trong các cửa hàng bán lẻ điện thoại di động.

Ngoài ra, chúng ta có thể phát triển thêm bài toán bằng cách thu nhập thêm thông tin về ý kiến của khách hàng về sản phẩm thông qua các đánh giá sản phẩm trực tiếp hoặc qua các trang web bán hàng của mình. Từ những đánh giá đó ta sẽ tiến hành phân tích, sử dụng các mô hình Deep Learning khác như Sentiment để phân tích ra được các tính năng, các yếu tố cần được cải thiện là gì từ đó thúc đẩy quá trình cải thiện điện thoại cũng như đẩy mạnh khả năng cạnh tranh thương mại của mình trên thị trường hiện nay.

TÀI LIỆU THAM KHẢO

Nguyễn Hưng, 2022, Deep Learning là gì? Tìm hiểu về Deep Learning từ A-Z, từ https://vietnix.vn/deep-learning-la-gi/

sklearn.ensemble.RandomForestClassifier — scikit-learn 1.2.2 documentation, truy cập 22-4 - 2023, từ https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

sklearn.svm.SVC — scikit-learn 1.2.2 documentation, truy cập 22 - 4 - 2023, từ https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Keras: Multilayer Perceptron (MLP) Example - Data Analytics, truy cập 22 - 4- 2023, từ https://vitalflux.com/keras-multilayer-perceptron-mlp-example/#Defining_the_MLP_Model_Architecture

PHỤ LỤC

Link github: https://github.com/vientriet/do-an-may-hoc