**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

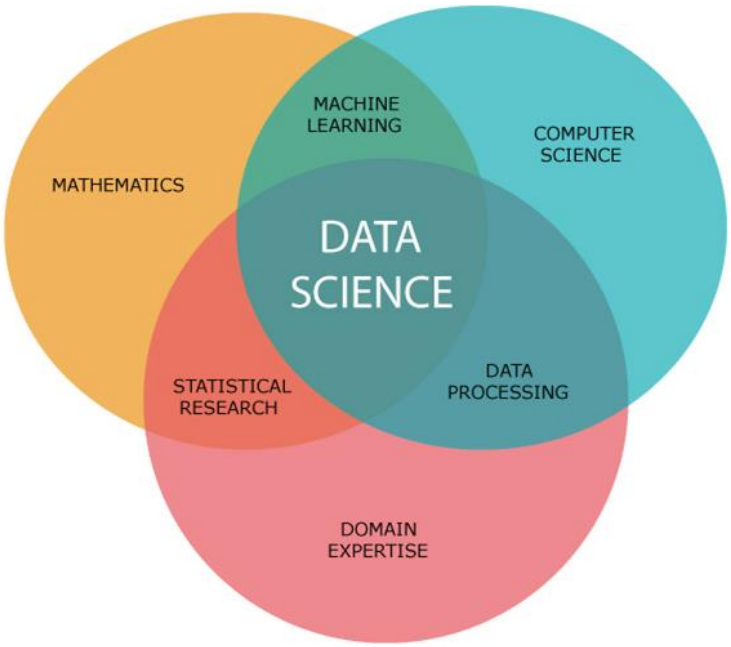
**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN:**

**KHOA HỌC DỮ LIỆU**

**BÁO CÁO ĐỒ ÁN:**

# DỰ ĐOÁN KHUNG GIÁ ĐIỆN THOẠI DI ĐỘNG




**GVLT: TRẦN TRUNG KIÊN**

# MỤC LỤC

## I. Thông tin nhóm:

| MSSV | Họ và tên | Liên lạc |
|---|---|---|
| 1612888 | Phan Minh Sơn | 1612888@student.hcmus.edu.vn |
| 1612892 | Trần Mạnh Thắng | 1612892@student.hcmus.edu.vn |

## II. Bài toán

Hiện nay, nhu cầu sử dụng smart phone ngày càng tăng cao, số lượng sản phẩm smart phone tung ra thị trường ngày càng nhiều. Để tăng tính cạnh tranh, các nhãn hàng sẽ cải thiện các thông số của điện thoại cũng như đưa ra giá cả hợp lí. Do đó, các công ty cần tham khảo giá thị trường các sản phẩm khác để định giá sản phẩm của mình dựa vào các chi tiết như CPU, RAM, Camera, thương hiệu ... Việc định giá được sản phẩm sẽ giúp nhà cung cấp đưa ra mức giá tốt hơn, doanh số bán hàng sẽ cao hơn. Đồng thời người tiêu dùng cũng sẽ biết được giá trị thực của sản phẩm mà không chạy theo giá trị thương hiệu, giúp chi tiêu sẽ hợp lí hơn.

Ở đồ án này, thực hiện sử dụng các thông tin về chi tiết của điện thoại để dự đoán khoảng giá của điện thoại.

Các thông tin – thuộc tính được sử dụng bao gồm 26 thuộc tính:

```
Bluetooth
Width
Height
Thick
Weight
Memory card
OS
SoC
Wi-Fi
GPU core
CPU core
SIM type
Number of SIM cards
USB type
USB version
Position tracking
Display size
Display resolution
Display color depth
Image resolution
Video resolution
FPS
Battery type
Battery power
RAM
Storage
```

Phân lớp dự đoán:

- 0: price<300$: mức giá thấp / rẻ.
- 1: 300$<=price<600$: mức giá trung bình
- 2: 600$<=price<900$: mức giá cao
- 3: 900$ <=price: mức giá rất cao

## III. Dữ liệu:

Dữ liệu được thu thập từ trang web https://www.devicespecifications.com. Ta sẽ thu thập bằng cách parse HTML.

Có hơn 11000 link sản phẩm, tuy nhiên chỉ có gần 3000 sản phẩm là có ghi bảng giá. Ta sẽ train với gần 3000 sản phẩm đó (nếu cần thiết sẽ tự thu thập giá trị của các sản phẩm không có bảng giá để tăng kích thước bộ train).

Cụ thể:
Thực hiện thu thập 2 loại data:

**Loại 1: Data về những dữ liệu cơ bản của điện thoại**
Tập ***dataset.csv***.
Số lượng, thuộc tính:

```
RangeIndex: 2811 entries, 0 to 2810
Data columns (total 21 columns):
Brand            2811 non-null object
Name             2811 non-null object
Dim              2811 non-null object
Weight           2811 non-null object
Bat              2811 non-null object
Blue             2811 non-null object
CPU              2811 non-null object
Cam              2811 non-null object
Cores            2811 non-null object
Dis              2811 non-null object
GPU              2811 non-null object
Memory card      2811 non-null object
OS               2811 non-null object
Pos              2811 non-null object
RAM              2811 non-null object
SIM              2811 non-null object
SoC              2811 non-null object
Storage          2811 non-null object
USB              2811 non-null object
Wi-Fi            2811 non-null object
Average price    2811 non-null object
```

**Loại 2: Data chi tiết về điện thoại**
Các tập ***o1.csv*** đến ***o4.csv***.
Số lượng mẫu: 2857
Số lượng thuộc tính : 100

Do chi tiết điện thoại quá nhiều thuộc tính nên ta chỉ chọn lựa lại các thuộc tính để thực hiện bài toán. Hầu hết các tính nằm trong loại 1.

Cụ thể, dữ liệu sau khi tiền xử lý ở file mobile_dataset.csv bao gồm 26 thuộc tính dự đoán và 1 thuộc tính phân lớp (Đã tình bày ở mục I).

Dữ liệu được thu thập và chuẩn hóa một cách hợp lý phục vụ cho quá trình học và kiểm thử (Phần chuẩn hóa dữ liệu được trình bày cụ thể trong Preprocessing.ipynb):
*1- 2 thuộc tính Brand, Name sẽ bị loại bỏ.*
Giải thích: Brand và Name có ảnh hưởng đến giá của sản phẩm, song, nó không thể làm giá quá chênh lệch giữa các nhãn hiệu khi cùng cấu hình. Do đó, nó không gây ảnh hưởng đến loại giá cần dự đoán.

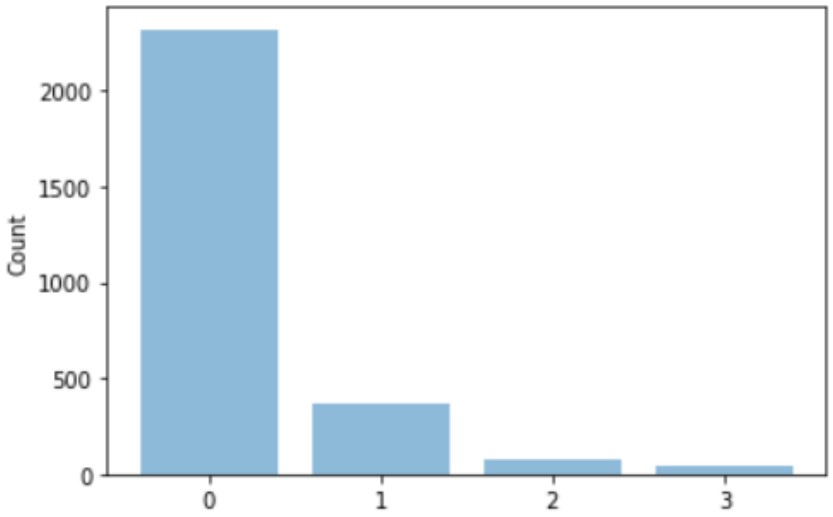
*2- Loại bỏ các thuộc tính có tỉ lệ giá trị thiều > 20%*

*3- PriceRange được chuẩn hóa như đã nêu ở phần 1*

*4- Đối với dữ liệu thiếu:*
- Dữ liệu dạng numeric sẽ sử dụng mean điền giá trị thiếu.
- Dữ liệu dạng categorical, sử dụng mode để điều giá trị thiếu

(được xử lý ở phần đầu file train-and-test-with-some-models.ipynb).

Như vậy, kết quả cuối cùng thu được, dùng để học và kiểm thử như sau:
*Tổng số mẫu: 2811 mẫu.*
*Tổng số thuộc tính để dự đoán: 26 thuộc tính (đã trình bày ở phần I).*
*Thuộc tính phân lớp: price_range.*
Phân bố của các lớp :



## IV. Quá trình thực hiện và kết quả:

Đối với bộ dữ liệu đã xử lý, thực hiện chia 2 tập train và test với tỷ lệ 4:1. Sử dụng các thuật toán khác nhau do thư viện sklearn hỗ trợ.

### MLP Classifier - Multi-layer Perceptron classifier

```
[26] # MLP Classifier
     full_pipeline = Pipeline(steps = [('preprocess_pipeline', preprocess_pipeline),
                                       ('NeuralNet', MLPClassifier(hidden_layer_sizes=(20), activation='tanh',
                                                                   solver='lbfgs', random_state=0, max_iter=1000))])

     train_accuracy = []
     test_accuracy = []
     alphas = [0.1, 0.5, 1, 10, 100, 1000]

     for alpha in alphas:
         full_pipeline.set_params(NeuralNet__alpha = alpha)
         full_pipeline.fit(train_X_df, train_Y_df)
         train_accuracy.append(full_pipeline.score(train_X_df, train_Y_df) * 100)
         test_accuracy.append(full_pipeline.score(test_X, test_Y) * 100)

     print(train_accuracy)
     print(test_accuracy)
```

```
[95.01779359430606, 97.59786476868328, 96.97508896797153, 93.41637010676158, 86.38790035587188, 82.6067615658363]
[84.90230905861456, 85.96802841918296, 86.32326820603907, 86.50088809946715, 87.38898756660745, 82.59325044404974]
```

Thực hiện chạy với tham số alpha thay đổi. Kết quả có thấy kết quả train, test khá cao. Kết quả đáng ghi nhận là: train_acr =97.598 và test_acr=85.968

### Logistic Regression

```
[27] # Logistic Regression
     full_pipeline = Pipeline(steps = [('preprocess_pipeline', preprocess_pipeline),
                                       ('LogReg', LogisticRegression(multi_class = 'multinomial', solver = 'sag',  max_iter = 10000))])

     full_pipeline.fit(train_X_df, train_Y_df)
     print('Train accuracy: ', full_pipeline.score(train_X_df, train_Y_df) * 100)
     print('Test accuracy: ', full_pipeline.score(test_X, test_Y) * 100)
```

```
Train accuracy:  88.79003558718861
Test accuracy:  87.2113676731794
```

### Decision Tree

```
[28] # Deision Tree
     full_pipeline = Pipeline(steps = [('preprocess_pipeline', preprocess_pipeline),
                                       ('Decision_Tree', DecisionTreeClassifier(random_state=101))])

     full_pipeline.fit(train_X_df, train_Y_df)
     print('Train accuracy: ', full_pipeline.score(train_X_df, train_Y_df) * 100)
     print('Test accuracy: ', full_pipeline.score(test_X, test_Y) * 100)
```

```
Train accuracy:  99.91103202846975
Test accuracy:  82.59325044404974
```

## Random Forest

```
[30] # Random Forest
     full_pipeline = Pipeline(steps = [('preprocess_pipeline', preprocess_pipeline),
                                       ('Random_Forest', RandomForestClassifier(n_estimators = 100,
                                                                                random_state=101, criterion = 'entropy', oob_score = True))])

     full_pipeline.fit(train_X_df, train_Y_df)
     print('Train accuracy: ', full_pipeline.score(train_X_df, train_Y_df) * 100)
     print('Test accuracy: ', full_pipeline.score(test_X, test_Y) * 100)
```

```
Train accuracy:  99.91103202846975
Test accuracy:  88.45470692717583
```

## KNN

```
[31] # KNN
     full_pipeline = Pipeline(steps = [('preprocess_pipeline', preprocess_pipeline),
                                       ('KNN', KNeighborsClassifier(n_neighbors=3))])

     full_pipeline.fit(train_X_df, train_Y_df)
     print('Train accuracy: ', full_pipeline.score(train_X_df, train_Y_df) * 100)
     print('Test accuracy: ', full_pipeline.score(test_X, test_Y) * 100)
```

```
Train accuracy:  89.72419928825623
Test accuracy:  83.30373001776199
```

*Nhận xét: Thuật toán DecisionTree và Random Forest đạt các kết quả tốt nhất.*

Ta thực hiện tùy chỉnh tham số trên Random Forest để đạt kết quả tốt nhất.

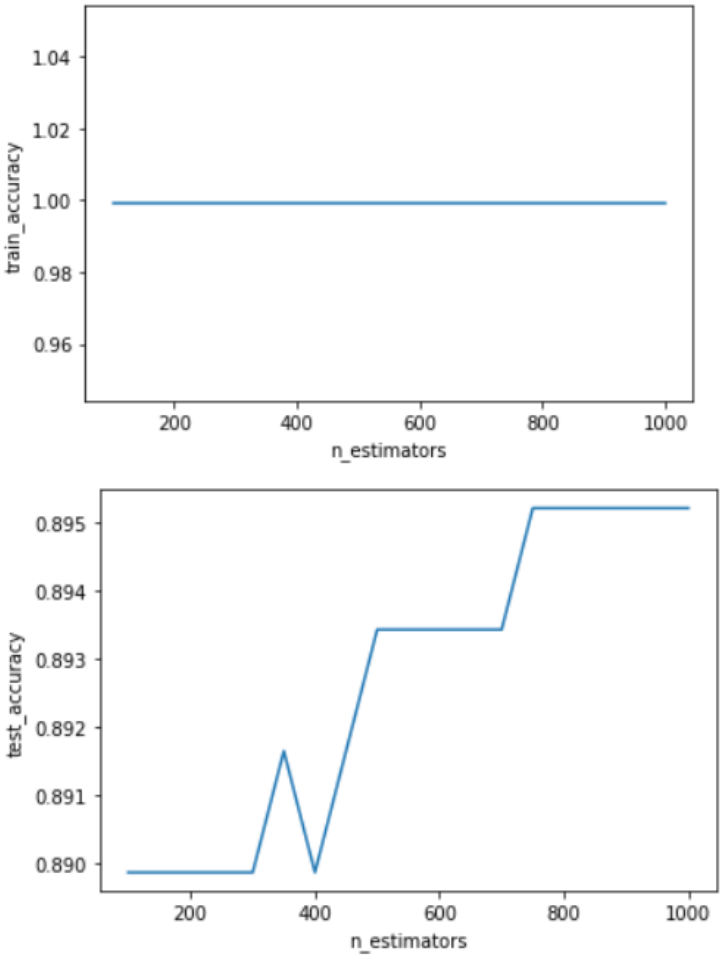
```
n_es=[x for x in range(100,500,50)]
criterions=['entropy','gini']
max_f=['auto','log2',None]
#print(n_es)
#n_estimators = 100,random_state=101, criterion = 'entropy'
for criterion in criterions:
  print(criterion)
  for f in max_f:
    print(f)
    for n in n_es:
      print(n)
      for r in range(0,100):
        full_pipeline = Pipeline(steps = [('preprocess_pipeline', preprocess_pipeline),
                                  ('Random_Forest', RandomForestClassifier(oob_score = True,random_state=40))])
        full_pipeline.set_params(Random_Forest__n_estimators=n,Random_Forest__criterion=criterion,
                                 Random_Forest__max_features=f,Random_Forest__random_state=r)
        full_pipeline.fit(train_X_df, train_Y_df)
        print('Train accuracy: ', full_pipeline.score(train_X_df, train_Y_df) * 100)
        print('Test accuracy: ', full_pipeline.score(test_X, test_Y) * 100)
```

Kết quả tốt nhất thu được với bộ tham số thay đổi là:
*criterion='entropy', n_estimators=300,*
*random_state=33, max_features=None*
là:
*Train=99.91103202846975*
*Test=89.69804618117229*

Nhận xét, qua thực nghiệm cho thấy:

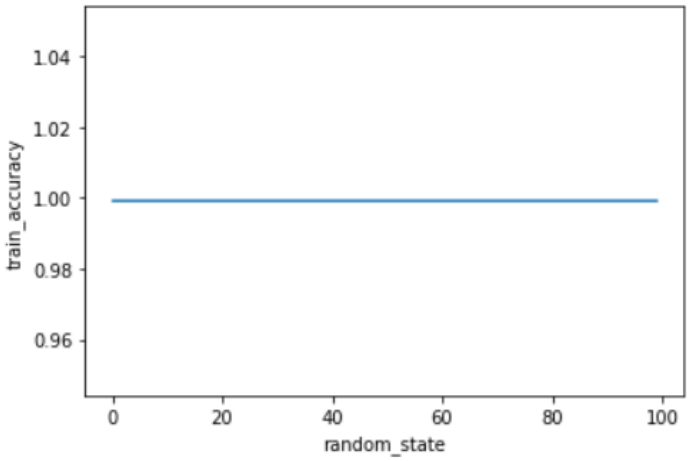
- *criterion = entropy* cho kết quả tốt hơn so với gini.
- *max_features= None* đạt kết quả tốt nhất.
- *n_estimators* và *random_state* có ảnh hưởng đến độ chính xác.
- Độ chính xác trên tập train ít thay đổi, độ chính xác trên tập test thay đổi nhiều hơn.

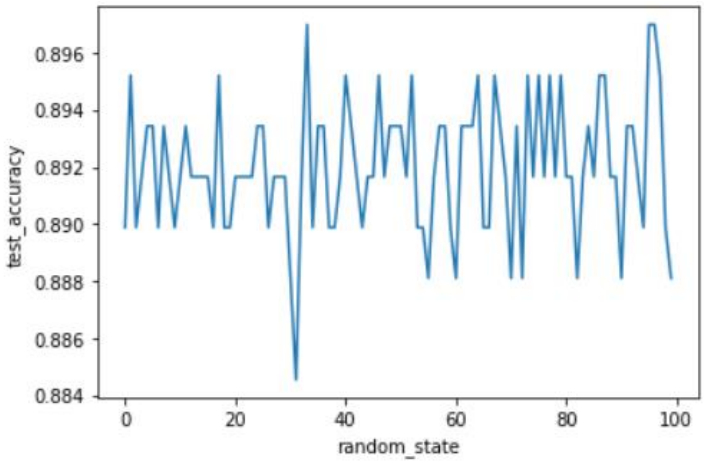
Ảnh hưởng của n_estimators:




*n_estimators tăng: train_accuracy gần như không đổi, test_accuracy có chiều hướng tăng và hội tụ.*

Ảnh hưởng của random_state:




*random_state tăng: train_accuracy gần như không đổi, test_accuracy có biến thiên nhiều.*

## V. Phân công và đánh giá:

Nhận xét về quá trình thực hiện được:

Về thu thập dữ liệu:

- Bài toán chưa thực sự thực tế.
- Nguồn dữ liệu khá khó tìm do các trang thương mại thường không đồng bộ nhiều về thuộc tính sản phẩm.
- Dữ liệu thô được lấy dễ bởi html request nhưng do có vấn đề với bộ ba giá trị (RAM, storage, price) -> giải pháp lấy giá trị trung bình : giải pháp còn chưa thỏa đáng.
- Xử lý dữ liệu thiếu: sử dụng mean cho numeric và mode cho categorical: giải pháp chưa thỏa đáng.

Về thực hiện học và kiểm thử:

- Sử dụng được nhiều thuật toán classification lên data, cho kết quả khá tốt.
- Tuy nhiên, do ảnh hưởng của dữ liệu nên kết quả trên tập test không đạt được đến 90%.

Phân công:

| **Công việc** | **Người thực hiện** | **Đánh giá** |
|---|---|---|
| Thu thập dữ liệu – thông tin cơ bản | Trần Mạnh Thắng | HT |
| Thu thập dữ liệu – Thông tin chi tiết | Phan Minh Sơn | HT |
| Chuẩn hóa dữ liệu | Trần Mạnh Thắng | HT |
| Học và kiểm thử | Trần Mạnh Thắng | HT |
| Thay đổi tham số để tối ưu kết quả | Phan Minh Sơn | HT |
| Báo cáo, Slide | Phan Minh Sơn | HT |

## VI. Tài liệu tham khảo:

https://scikit-learn.org/stable/index.html