Mục lục

Mục lục các hình

Phần 1: Thuật toán J48 Decision Tree

1.1. Lý thuyết thuật toán J48 Decision Tree

1.1.1. Cây quyết định

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật (series of rules) hay cây quyết định là cây mà có mỗi nút sẽ biểu diễn một tính chất nào đó trong một tập các tính chất với mỗi nút sẽ sinh ra các nhánh khác nhau để biểu diễn một quy luật và mỗi lá sẽ hiển thị các kết quả của các nhánh.

1.1.2. Giới thiệu về thuật toán J48

J48 là một phần mở rộng của ID3 có tên đầy đủ là weka.classifier.trees.J48 là một triển khai Java mã nguồn mở của thuật toán C4.5 dùng xây dựng cây quyết định trong Weka dưới dạng Classification.

C4.5 là một thuật toán được sử dụng để tạo cây quyết định được phát triển bởi Ross Quinlan .Classification là quá trình xây dựng mô hình các lớp từ một tập hợp các bản ghi có chứa nhãn lớp. Weka là tên viết tắt của Waikato Environment for Knowledge Analysis là một bộ phần mềm học máy được Đại học Waikato, New Zealand phát triển bằng Java.

1.1.3. Các bước cơ bản trong thuật toán

B1: Dán nhãn cho các trường hợp các thể hiện thuộc cùng một lớp với cùng một lớp.

B2: Thông tin tiềm năng được tính cho mọi thuộc tính, được đưa ra bởi một thử nghiệm trên thuộc tính. Sau đó, mức tăng thông tin được tính toán sẽ dẫn đến kết quả kiểm tra thuộc tính.

B3: Thuộc tính tốt nhất được tìm thấy trên cơ sở tiêu chí lựa chọn hiện tại và thuộc tính đó được chọn để phân nhánh.

1.1.4. Ví dụ

Giả sử nhóm em có một tập dữ liệu Iris thông tin về các loài hoa bao gồm 4 thuộc tính **sepallength**, **sepalwidth**, **petallength**, **petalwidth** và thuộc 3 phân loại có giá trị là **Iris-setosa**, **Iris-versicolor**, **Iris-virginica**.

sepalleng	sepalwidth	petallength	petalwidth	class
5	2	3.5	1	Iris-versicolor
6	2.2	4	1	Iris-versicolor
6.2	2.2	4.5	1.5	Iris-versicolor
6	2.2	5	1.5	Iris-virginica
4.5	2.3	1.3	0.3	Iris-setosa
5.5	2.3	4	1.3	Iris-versicolor
6.3	2.3	4.4	1.3	Iris-versicolor
5	2.3	3.3	1	Iris-versicolor
4.9	2.4	3.3	1	Iris-versicolor
5.5	2.4	3.8	1.1	Iris-versicolor
5.5	2.4	3.7	1	Iris-versicolor

Hình 1: Tập dữ liệu Iris

Các thuộc tính của hoa	Giá trị
sepallength	5
sepalwidth	2
petallength	3.5
petalwidth	1
Loại	?

Hình 2 : Nhận dạng loại hoa dựa trên các thuộc tính

Mục tiêu với tập dữ liệu được đưa ra ở trên thì việc sau khi Training dữ liệu và chạy thử thuật toán sẽ phân loại với giá trị này thì cây sẽ được phân lớp theo loại cây nào trong 3 phân loại (**Iris-setosa**, **Iris-versicolor**, **Iris-virginica**).

```
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)
```

Hình 3 : Tập dữ liệu thu được sau khi thuật toán kết thúc

Với những dữ liệu thu được thì ta có thể dễ dàng hiểu như sau:

- Nếu petalwidth <= 0.6 sẽ là hoa Iris-setosa
- Nếu 0.6 < pentallwidth <=1.7:
  - petallength <= 4.9 sẽ là hoa Iris-versicolor.
  - petallength > 4.9 và pentalwidth <= 1.5 sẽ là hoa Iris-virginica.
  - petallength > 4.9 và pentalwidth > 1.5 sẽ là hoa Iris-versicolor.
- Nếu pentallwidth > 1.7: sẽ là hoa Iris-virginica

1.1.5. Cắt tỉa cây

Một số trường hợp có mặt trong tất cả các tập dữ liệu không được xác định rõ và khác với các trường hợp khác trên vùng lân cận. Việc phân loại được thực hiện trong các trường hợp của tập training set và cây được hình thành. Việc cắt tỉa được thực hiện để giảm các lỗi phân loại đang được tạo ra bởi sự chuyên môn hóa trong tập huấn luyện. Cắt tỉa được thực hiện cho việc khái quát hóa của cây

1.1.6. Các bước được thực hiện để khai thác dữ liệu trong WEKA

B1: Data pre-processing and visualization

B2: Attribute selection

B3: Classification (Decision trees)

B4: Prediction (Nearest neighbour)

B5: Model evaluation

B6: Clustering (Cobweb, K-means)

B7: Association rules

1.1.7. Thuật toán J48 và ứng dụng trong thực tế

-Dùng để khai thác dữ liệu (dự báo năng suất cây trồng ,dự đoán thời tiết …)

-Ứng dụng trong ngành y học như (Dự đoán bệnh tiểu đường ,bệnh…)

- Quản lý quan hệ khách hàng

- Mạng lưới thần kinh nhân tạo.

1.2. Demo thuật toán J48

1.2.1. Các thư viện sử dụng và Dataset Iris

❖ Các thư viện sử dụng

```
import weka.core.jvm as jvm
from weka.classifiers import Classifier
from weka.classifiers import Evaluation
from weka.core.classes import Random
from weka.core.converters import Loader
import weka.plot.graph as graph
import weka.plot.classifiers as plotcls
```

Thư viện	Chức năng
weka.core.jvm	Dùng để khởi động môi trường Java ảo (Java virtual machine)
Classifier	Hỗ trợ các thuật toán Decision Tree J48,Naivy bayes,Support vector machines
Evaluation	Xuất ra các kết quả sau khi thuật toán đã thực hiện xong (Precision, recall)
Random	Random seed cho phân tách dữ liệu xác thực chéo
Loader	Hỗ trợ load các dữ liệu từ file hệ thống
Weka.plot.graph	Hỗ trợ vẽ hình

❖ Dataset Iris

Stt	Thuộc tính	Các giá trị
1	sepallength	Real
2	sepalwidth	Real
3	petallength	Real
4	petalwidth	Real
5	class	Iris-setosa
		Iris-versicolor
		Iris-virginica

❖ Ý nghĩa dataset

Stt	Thuộc tính	Ý nghĩa	
1	sepallength	Chiều dài đài hoa	
2	sepalwidth	Chiều rộng đài hoa	
3	petallength	Chiều dài cánh hoa	
4	petalwidth	Chiều rộng cánh hoa	
5	class	Loại hoa	setosa
			versicolor
			virginica

- Có tất cả 150 dòng dữ liệu
- Thuộc tính phân lớp thuộc vào thuộc tính 5 (class)

1.2.2. Chạy thuật toán

```
jvm.start()
data_dir = "C:/Users/Huyvo/Downloads/"
loader = Loader(classname="weka.core.converters.ArffLoader")
data = loader.load_file(data_dir + "iris.arff")
data.class_is_last()
```

Đoạn code	Ý nghĩa
jvm.start()	Dùng để khởi động môi trường Java ảo
data_dir	Chỉ ra đường dẫn chưa file dữ liệu
loader	Gọi thư viện load file .arff
loader.load_file()	Lấy file từ đường dẫn
data.class_is_last()	Gắn thuộc tính cho class

```
# classifier = Classifier(classname="weka.classifiers.trees.J48")
# classifier.set_property("confidenceFactor",types.double_to_float(0.3))
classifier = Classifier(classname="weka.classifiers.trees.J48",
                        options=["-C", "0.3"])
classifier.build_classifier(data)
evaluation = Evaluation(data)
#evaluation.crossvalidate_model(classifier,data,10,Random(42))
evaluation.evaluate_train_test_split(classifier, data,30, Random(1))
```

Đoạn code	Ý nghĩa
Classifier(classname="weka.classifiers.trees.J48")	Phân lớp được quyết định theo thuật toán cây quyết định J48.
options = [“-C”,”0.3”]	Sẽ tương đương với classifier.set_property ("confidenceFactor",types.double_to_float(0.3)) Trong đó confidenceFactor là độ tin cậy để cắt tỉa cây với giá trị là 0.3 (giá trị mặc định là 0.25)
classifier.build_classifier(data)	Thực hiện phân lớp dữ liệu dựa trên thuật toán J48 đã chọn
evaluation.crossvalidate_model (classifier,data,10,Random(42))	Đánh giá mô hình dữ liệu theo phương pháp k-fold cross validation (K=10) với ý nghĩa các tập dữ liệu sẽ được chia ra làm 10 phần bằng nhau (1 phần test, 9 phần train)
evaluation.evaluate_train_test_split (classifier, data,30, Random(1))	Đánh giá mô hình dữ liệu bằng cách chia dữ liệu tùy ý, ở trường hợp này 30% dữ liệu là test và 70% dữ liệu là train.

❖ Kết quả của thuật toán:

Với phương pháp Crossvalidate:

```
Correctly Classified Instances         142               94.6667 %
Incorrectly Classified Instances         8                5.3333 %
Kappa statistic                          0.92
Mean absolute error                      0.0439
Root mean squared error                  0.185
Relative absolute error                  9.869  %
Root relative squared error             39.2433 %
Total Number of Instances              150

pctCorrect: 94.6666666667
incorrect: 8.0
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.980     0.000     1.000      0.980     0.990      0.985    0.990     0.987     Iris-setosa
                 0.940     0.050     0.904      0.940     0.922      0.882    0.944     0.884     Iris-versicolor
                 0.920     0.030     0.939      0.920     0.929      0.895    0.953     0.871     Iris-virginica
Weighted Avg.    0.947     0.027     0.948      0.947     0.947      0.920    0.962     0.914
```

Hình 4 : Kết quả đánh giá mô hình theo phương pháp k-fold Cross Validation

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

Hình 5 : Ma trận thể hiện các phân lớp đúng và sai

Class	Ý nghĩa
a	49 dòng dữ liệu được dự đoán đúng là hoa Iris-setosa và 1 dòng dữ liệu bị dự đoán nhầm sang lớp b (Iris-setosa)
b	47 dòng dữ liệu được dự đoán đúng là hoa Iris-versicolor và 3 dòng dữ liệu bị dự đoán nhầm sang lớp c (Iris-virginica)
c	46 dòng dữ liệu được dự đoán đúng là hoa Iris-virginica và 4 dòng dữ liệu bị dự đoán nhầm sang lớp b (Iris-versicolor)

	Số mẫu	Tỉ lệ
Phân lớp đúng	142	94.667%
Phân lớp sai	8	5.333%
Tổng	150	100%

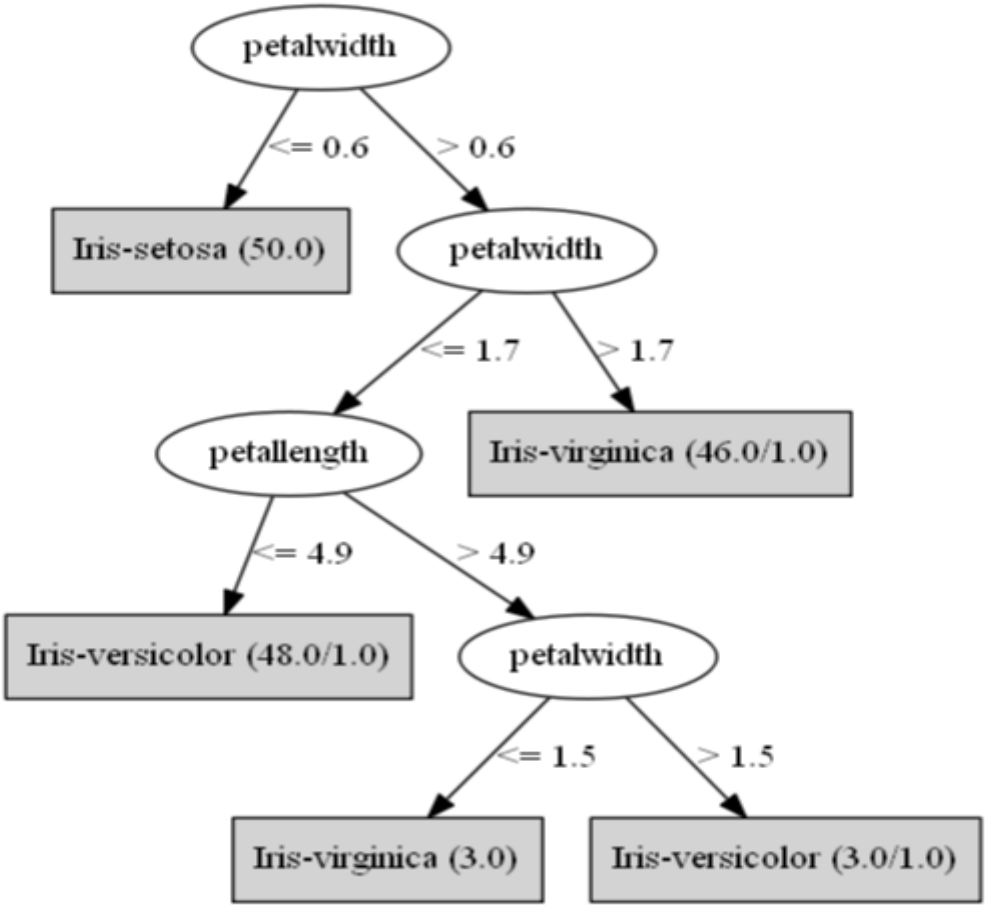
```
J48 pruned tree
------------------

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :  9
```

Hình 6 : Cây quyết định sẽ có số lá là 5 và với kích thước của cây là 9



Hình 7 : Cây quyết định

Với phương pháp Train test split:

- Với 65% mẫu là dữ liệu test và 35% mẫu là dữ liệu train

```
Correctly Classified Instances          51               96.2264 %
Incorrectly Classified Instances         2                3.7736 %
Kappa statistic                          0.9429
Mean absolute error                      0.0388
Root mean squared error                  0.1549
Relative absolute error                  8.7279 %
Root relative squared error             32.8523 %
Total Number of Instances               53

pctCorrect: 96.2264150943
incorrect: 2.0
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.061    0.909      1.000    0.952      0.924    0.970     0.909     Iris-versicolor
                 0.882    0.000    1.000      0.882    0.938      0.914    0.967     0.936     Iris-virginica
Weighted Avg.    0.962    0.023    0.966      0.962    0.962      0.944    0.978     0.945

=== Confusion Matrix ===

  a  b  c   <-- classified as
 16  0  0 |  a = Iris-setosa
  0 20  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

Hình 8 : Kết quả đánh giá mô hình theo phương pháp Train test split lần 1

	Số mẫu	Tỉ lệ
Phân lớp đúng	55	91.667%
Phân lớp sai	5	8.333%
Tổng	60	

- Với 80% mẫu là dữ liệu test và 20% mẫu là dữ liệu train

```
Correctly Classified Instances          30               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0105
Root mean squared error                  0.0166
Relative absolute error                  2.3684 %
Root relative squared error              3.5306 %
Total Number of Instances               30

pctCorrect: 100.0
incorrect: 0.0
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-versicolor
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-virginica
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

  a  b  c   <-- classified as
 11  0  0 |  a = Iris-setosa
  0 10  0 |  b = Iris-versicolor
  0  0  9 |  c = Iris-virginica
```

Hình 9 : Kết quả đánh giá mô hình theo phương pháp Train test split lần 2

	Số mẫu	Tỉ lệ
Phân lớp đúng	30	100%
Phân lớp sai	0	0%
Tổng	30	100%

- Với 30% mẫu dữ liệu là test và 70% mẫu dữ liệu là train

```
Correctly Classified Instances         100               95.2381 %
Incorrectly Classified Instances         5                4.7619 %
Kappa statistic                          0.9287
Mean absolute error                      0.0497
Root mean squared error                  0.1823
Relative absolute error                 11.1905 %
Root relative squared error             38.6786 %
Total Number of Instances              105

pctCorrect: 95.2380952381
incorrect: 5.0
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.068    0.865      1.000    0.928      0.898    0.966     0.865     Iris-versicolor
                 0.872    0.000    1.000      0.872    0.932      0.900    0.936     0.919     Iris-virginica
Weighted Avg.    0.952    0.021    0.959      0.952    0.952      0.932    0.966     0.929
```

Hình 10 : Kết quả đánh giá mô hình theo phương pháp Train test split lần 3

	Số mẫu	Tỉ lệ
Phân lớp đúng	100	95.2381%
Phân lớp sai	5	4.7619%
Tổng	105	100%

- Với 10% mẫu dữ liệu là test và 90% mẫu dữ liệu là train

```
Correctly Classified Instances         124               91.8519 %
Incorrectly Classified Instances        11                8.1481 %
Kappa statistic                          0.8776
Mean absolute error                      0.0909
Root mean squared error                  0.2177
Relative absolute error                 20.4444 %
Root relative squared error             46.188  %
Total Number of Instances              135

pctCorrect: 91.8518518519
incorrect: 11.0
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.956     0.000     1.000      0.956    0.977      0.967    0.978     0.970     Iris-setosa
                 0.978     0.112     0.818      0.978    0.891      0.835    0.933     0.808     Iris-versicolor
                 0.818     0.011     0.973      0.818    0.889      0.848    0.947     0.883     Iris-virginica
Weighted Avg.    0.919     0.042     0.929      0.919    0.919      0.883    0.952     0.887

=== Confusion Matrix ===

  a  b  c   <-- classified as
 43  2  0 |  a = Iris-setosa
  0 45  1 |  b = Iris-versicolor
  0  8 36 |  c = Iris-virginica
```

Hình 11 : Kết quả đánh giá mô hình theo phương pháp Train test split lần 4

	Số mẫu	Tỉ lệ
Phân lớp đúng	124	91.8519%
Phân lớp sai	11	8.1481%
Tổng	135	100%

- Với 5% dữ liệu là test và 95% dữ liệu là train

```
Correctly Classified Instances          63               44.0559 %
Incorrectly Classified Instances        80               55.9441 %
Kappa statistic                          0.1667
Mean absolute error                      0.3427
Root mean squared error                  0.5231
Relative absolute error                 77.0979 %
Root relative squared error            110.9597 %
Total Number of Instances              143

pctCorrect: 44.0559440559
incorrect: 80.0
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.000     0.000     ?          0.000   ?          ?        0.840     0.620     Iris-setosa
               0.489     0.583     0.291      0.489   0.365      -0.089   0.453     0.310     Iris-versicolor
               0.851     0.250     0.625      0.851   0.721      0.568    0.801     0.581     Iris-virginica
Weighted Avg.  0.441     0.274     ?          0.441   ?          ?        0.700     0.505

=== Confusion Matrix ===

  a  b  c   <-- classified as
  0 49  0 |  a = Iris-setosa
  0 23 24 |  b = Iris-versicolor
  0  7 40 |  c = Iris-virginica
```

Hình 12 : Kết quả đánh giá mô hình theo phương pháp Train test split lần 5

	Số mẫu	Tỉ lệ
Phân lớp đúng	63	44.0559%
Phân lớp sai	80	55.9441%
Tổng	143	100%

1.3. Đánh giá

❖ Với phương pháp K-folder cross validation

- Để tránh việc trùng lặp giữa các dữ liệu với nhau trong quá trình kiểm thử thì phương pháp này chia dữ liệu ban đầu thành *k* tập dữ liệu nhỏ có kích thước bằng hoặc gần bằng nhau trong đó sẽ có 1 mẫu chứa dữ liệu là testing và *k-1* mẫu chứa dữ liệu là training.
- Do việc chia dữ liệu training là chủ yếu nên việc sử dụng phương pháp này sẽ cho ra kết quả khả thi.

❖ Với phương pháp Train test split

- Với 2 lần chạy đầu tiên thì việc phân chia dữ liệu test > 65% sẽ cho kết quả phân loại cao, tuy nhiên với dữ liệu test càng cao thì đồng nghĩa với việc tổng số mẫu sẽ bị giảm nên hiểu quả phân lớp sẽ không được xem là hiểu quả cao nhất.

- Với lần 3 lần chạy tiếp theo thì có thể thấy nếu chia dữ liệu test ở giá trị 5% thì việc phân lớp đúng sẽ giảm đi đáng kể, còn ở 10% và 30% vẫn cho kết quả phân lớp tốt. Do giữa 10% và 30% đều vẫn cho kết quả lớn hơn 90% tuy nhiên với tổng số mẫu để thực hiện cho việc phân lớp ở 10% là nhiều hơn và vẫn cho được kết quả tốt nên phương pháp Train test với tập dữ liệu Iris sẽ đạt hiệu quả phân lớp cao nhất với tỉ lệ dữ liệu test là 10%

Phần 2 : Random Forest

2.1. Lý thuyết

2.1.1. Giới thiệu Random Forest

Random Forest được xuất hiện lần đầu tiên vào năm 1995, sau đó kết hợp với kỹ thuật lựa chọn các thuộc tính ngẫu nhiên của Leo Breiman năm 1996. Và đến năm 2001 Leo Breiman xây dựng thuật toán Random Forest có bổ sung thêm một lớp ngẫu nhiên để phân lớp.

Ngoài việc mỗi cây sử dụng các mẫu dữ liệu khác nhau, rừng ngẫu nhiên được thay đổi để xây dựng các cây phân loại và hồi quy khác nhau. Các gói thư viện cài đặt thuật toán Random Forest đầu tiên được xây dựng bằng ngôn ngữ Fortran bởi Leo Breiman và Cutler.

Random Forest được mô hình hóa như tập các cây phân lớp. Tuy nhiên Random Forest sử dụng các mẫu ngẫu nhiên cho các cây cũng như việc chọn lựa thuộc tính ngẫu nhiên khi phân chia cây. Thuật toán Random Forest tỏ ra chính xác và nhanh hơn khi train trên không gian dữ liệu lớn với nhiều thuộc tính, việc sử dụng kết quả dự đoán của cả tất cả các cây trong rừng khi phân lớp hoặc hồi quy giúp cho kết quả thuật toán chính xác hơn.

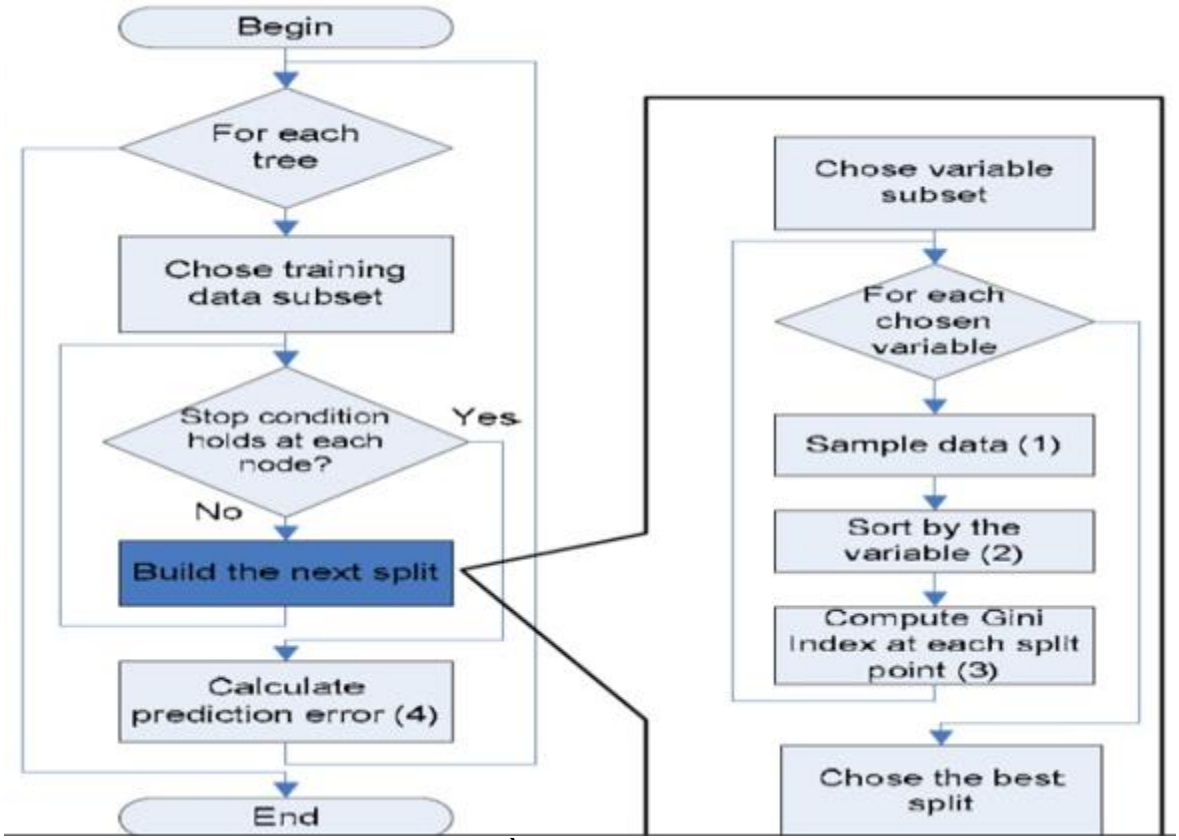
2.1.2. Lý thuyết Random Forest

Random Forest là một thuật toán học tập có giám sát, là một tập hợp của hai hay nhiều cây quyết định.Việc xây dựng nhiều cây quyết định và hợp nhất chúng lại với nhau để có được dự đoán chính xác và ổn định hơn. Một lợi thế lớn của Random Forest là nó có thể được sử dụng cho cả các vấn đề phân loại và hồi quy, tạo thành phần lớn các hệ thống máy học hiện tại. Số lượng các cây phân lớp trong rừng là không hạn chế và thuật toán sử dụng kết quả dự đoán của tất cả cây trong rừng làm kết quả cuối cùng của thuật toán.

2.1.3. Thuật toán Random Forest

- Đặt số lượng các trường hợp training là N và số lượng biến phân loại là M.
- Số m của các biến đầu vào được sử dụng để xác định quyết định tại một nút của cây (m nên nhỏ hơn M).
- Chọn một bộ huấn luyện cho cây này bằng cách chọn N lần thay thế từ tất cả N trường hợp đào tạo có sẵn. Sử dụng phần còn lại của các trường hợp để ước tính lỗi của cây, bằng cách dự đoán các lớp của chúng.
- Đối với mỗi nút của cây, chọn ngẫu nhiên m biến để dựa vào quyết định tại nút đó. Tính toán phân chia tốt nhất dựa trên các biến m này trong tập huấn luyện.
- Mỗi cây được trồng đầy đủ và không được cắt tỉa

2.1.4. flowchart random forest



Hình 13 : Sơ đồ flowchart random forest

2.1.5. Hoạt động của thuật toán rừng ngẫu nhiên như sau

B1: Một hạt giống ngẫu nhiên được chọn sẽ lấy ra một cách ngẫu nhiên một bộ sưu tập mẫu từ bộ dữ liệu huấn luyện trong khi duy trì phân phối lớp.

B2: Với bộ dữ liệu được chọn này, một bộ thuộc tính ngẫu nhiên từ bộ dữ liệu gốc được chọn dựa trên các giá trị do người dùng xác định. Tất cả các biến đầu vào không được xem xét vì tính toán rất lớn và cơ hội phù hợp cao.

B3: Trong một tập dữ liệu trong đó M là tổng số thuộc tính đầu vào trong tập dữ liệu, chỉ các thuộc tính R được chọn ngẫu nhiên cho mỗi cây trong đó R <M

B4: Các thuộc tính từ bộ này tạo ra sự phân chia tốt nhất có thể bằng cách sử dụng chỉ số gini để phát triển mô hình cây quyết định. Quá trình lặp lại cho mỗi nhánh cho đến khi điều kiện kết thúc cho biết các lá là các nút quá nhỏ để phân chia.

2.1.6. Đặc điểm của Random Forest

2.1.6.1. Ưu điểm

- Thuật toán giải quyết tốt các bài toán có nhiều dữ liệu nhiễu, thiếu giá trị. Do cách chọn ngẫu nhiên thuộc tính nên các giá trị nhiễu, thiếu ảnh hưởng không lớn đến kết quả.
- Có những sự ước lượng nội tại như độ chính xác của mô hình phỏng đoán hoặc độ mạnh và liên quan giữa các thuộc tính(Out of bag)
- Dễ dàng thực hiện song song. Thay vì một máy thực hiện cả thuật toán, ta có thể sử dụng nhiều máy để xây dựng các cây sau đó ghép lại thành rừng.
- Các sai số được giảm thiểu do kết quả của Random Forest được tổng hợp thông qua nhiều cây phân lớp
- Nó tính toán các giá trị gần đúng giữa các cặp trường hợp có thể được sử dụng trong phân cụm, định vị các ngoại lệ hoặc (bằng cách chia tỷ lệ) cho các chế độ xem thú vị của dữ liệu
- Lỗi chung của một rừng các cây phân lớp phụ thuộc vào lỗi riêng của từng cây trong rừng cũng như mối tương quan giữa các cây.

2.1.6.2. Nhược điểm

- Dữ liệu huấn luyện cần được đa dạng hóa và cân bằng về số nhãn lớp. Việc không cân bằng nhãn lớp khiến kết quả dự đoán của thuật toán có thể lệch về số đông nhãn lớp.
- Thời gian huấn luyện của rừng có thể kéo dài tùy số cây và số thuộc tính phân chia.

2.1.7. Ứng dụng thực tế

Thuật toán rừng ngẫu nhiên được sử dụng trong rất nhiều lĩnh vực khác nhau, như Ngân hàng, Thị trường chứng khoán, Y học và Thương mại điện tử.

- Trong Ngân hàng, ví dụ, nó được sử dụng để phát hiện những khách hàng sẽ sử dụng dịch vụ của ngân hàng thường xuyên hơn những người khác và trả nợ đúng hạn. Trong miền này, nó cũng được sử dụng để phát hiện những khách hàng lừa đảo muốn lừa đảo ngân hàng.
- Trong tài chính, nó được sử dụng để xác định hành vi của một cổ phiếu trong tương lai.
- Trong lĩnh vực chăm sóc sức khỏe, nó được sử dụng để xác định sự kết hợp chính xác của các thành phần trong y học và để phân tích lịch sử y tế của bệnh nhân để xác định bệnh.
- Và cuối cùng, trong rừng ngẫu nhiên thương mại điện tử được sử dụng để xác định xem khách hàng có thực sự thích sản phẩm đó hay không.

2.2. Demo thuật toán Random Forest

2.2.1. Dataset Temps và các thư viện được sử dụng

❖ Dataset

Stt	Thuộc tính	Ý nghĩa	Các giá trị
1	**clump_thickness**	Độ dày của khối u	Int
2	**uniformity_of_cell_size**	Tính đồng nhất của kích thước tế bào	Int
3	**uniformity_of_cell_shape**	Tính đồng nhất của hình dạng tế bào	Int
4	**marginal_adhesion**	Bám dính biên	String
5	**single_epithelial_cell_size**	Kích thước tế bào biểu mô	Int
6	**bare_nuclei**	Hạch của tế bào	Int
7	**bland_chromatin**	Nhiễm sắc thể	Int
8	**normal_nucleoli**	Tế bào bình thường	Int
9	**mitosis**	Nguyên phân	Int

❖ Các thư viện sử dụng

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import export_graphviz
import pydot
import os
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

Thư viện	Chức năng
import pandas as pd	panda được sử dụng để thao tác dữ liệu
import numpy as np	numpy được sử dụng để chuyển đổi sang mảng
from sklearn.model_selection import train_test_split	sklearn được sử dụng để phân chia dữ liệu thành các tập dữ liệu training và testing sets
from sklearn.ensemble import RandomForestRegressor	Import thư viện RandomForest
from sklearn.tree import export_graphviz	Import thêm công cụ Graphviz để trực quan hóa mô hình
import pydot	Import thư viện hỗ trợ vẽ Graphiviz
import os	Import thư viện kết nối Path
import matplotlib.pyplot as plt	Import thư viện matplotlib để vẽ và sử dụng lệnh
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score	Import thêm công cụ tính classfication_report, matrix và accuracy

2.2.2. Chạy thuật toán

2.2.2.1. Load dữ liệu và hiển thị mô tả dataset temps

```
features = pd.read_csv('C:/Users/user/Downloads/Breast_Cancer.csv')
print features.describe()
```

Code	Ý nghĩa		
features = pd.read_csv('C:/Users/user/Downloads/Breast_Cancer.csv')	Load data set Breast_Cancer từ đường dẫn		
print features.describe()	Thống kê mô tả cho từng cột	count	Đếm số dòng
		mean	Độ lệch trung bình
		std	Độ lệch chuẩn
		min / max	Giá trị nhỏ nhất và lớn nhất
		25/50/75%	Giá trị nằm vùng 25/50/75%

	clump_thickness	uniformity_of_cell_size	...	mitosis	class
count	569.000000	569.000000	...	569.000000	569.000000
mean	4.539543	3.184534	...	1.637961	0.634446
std	2.896501	3.002236	...	1.773941	0.482009
min	1.000000	1.000000	...	1.000000	0.000000
25%	2.000000	1.000000	...	1.000000	0.000000
50%	4.000000	1.000000	...	1.000000	1.000000
75%	6.000000	5.000000	...	1.000000	1.000000
max	10.000000	10.000000	...	10.000000	1.000000

Hình 14 : Hiển thị mô tả dataset temps

2.2.2.2. Mã hóa dữ liệu chuỗi (Nếu có dữ liệu chuỗi)

```
features = pd.get_dummies(features)
print features.iloc[:,11:].head(5)
```

Code	Ý nghĩa
features = pd.get_dummies(features)	Mã hóa dữ liệu chuỗi của features (tách cột dataset có dữ liệu chuỗi thành các cột dataset riêng)
print features.iloc[:,11:].head(5)	Hiển thị 5 dòng đầu của dataset cột 11 trở đi

Ví dụ giá trị mẫu của dataset temps cột week có 6 values như hình vẽ

year	month	day	week	temp_2	temp_1	average	actual	forecast_r	forecast_a	forecast_u	friend
2016	1	1	Fri	45	45	45.6	45	43	50	44	29
2016	1	2	Sat	44	45	45.7	44	41	50	44	61
2016	1	3	Sun	45	44	45.8	41	43	46	47	56
2016	1	4	Mon	44	41	45.9	40	44	48	46	53
2016	1	5	Tues	41	40	46	44	46	46	46	41
2016	1	6	Wed	40	44	46.1	51	43	49	48	40
2016	1	7	Thurs	44	51	46.2	45	45	49	46	38
2016	1	8	Fri	51	45	46.3	48	43	47	46	34
2016	1	9	Sat	45	48	46.4	50	46	50	45	47
2016	1	10	Sun	48	50	46.5	52	45	48	48	49
2016	1	11	Mon	50	52	46.7	45	42	48	48	39
2016	1	12	Tues	52	45	46.8	49	44	50	45	61

Thì khi sử dụng pd.get_dummies(features) thì các dữ liệu chuỗi sẽ bị cắt khỏi dataset và tạo thành các cột feature mới gắn nối tiếp vào dataset (bảng dataset bên dưới là biểu thị các cột tiếp theo của dataset temps)

	week_Fri	week_Mon	week_Sat	week_Sun	week_Thurs	week_Tues	week_Wed
0	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0
2	0	0	0	1	0	0	0
3	0	1	0	0	0	0	0
4	0	0	0	0	0	1	0

2.2.2.3. Tách giá trị target tùy ý ra khỏi dataset

```
x= features.iloc[:,:-1]
labels = np.array(features['class'])
features= features.drop('class', axis = 1)
feature_list = list(features.columns)
features = np.array(features)
```

Code	Ý nghĩa
x= features.iloc[:,:-1]	X bao gồm toàn bộ feature trừ feature cuối là 'class'
labels = np.array(features['class'])	Tạo label cho feature có tên 'class'
features= features.drop('class', axis = 1)	Drop features ra khỏi dataset axis=1 ứng với 1 feature
feature_list = list(features.columns)	Lưu lại danh sách các cột
features = np.array(features)	Chuyển đổi sang mảng

2.2.2.4. Chia Train , test cho features và labels

```
train_features, test_features, train_labels, test_labels =\
    train_test_split(features, labels, test_size = 0.25, random_state = 42)
print('Training Features Shape:', len(train_features))
print('Testing Features Shape:', len(test_features))
```

Code	Ý nghĩa
train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size = 0.25, random_state = 42)	Chia train,test cho features và labels (features là các cột dataset dữ liệu , labels là cột target)
print('Training Features Shape:', len(train_features))	Đếm độ dài của train features
print('Testing Features Shape:', len(test_features))	Đếm độ dài của test features

```
('Training Features Shape:', 426)
('Testing Features Shape:', 143)
```

Hình 15 : Hiển thị giá trị training và Testing

2.2.2.5. Khởi tạo RandomForest và các hàm tính

```
rf = RandomForestRegressor(n_estimators = 20, max_depth = 3,  random_state = 42)
rf.fit(train_features, train_labels)
predictions = rf.predict(test_features)
print "confusion_matrix"
print(confusion_matrix(test_labels,predictions.round()))
print "classification_report"
print(classification_report(test_labels,predictions.round()))
print "accuracy_score"
print(accuracy_score(test_labels, predictions.round()))
```

Code	Ý nghĩa
rf = RandomForestRegressor(n_estimators = 20, max_depth = 3, random_state = 42)	Khởi tạo random forest gồm 20 cây quyết định và giới hạn độ sâu của cây đến cấp 3
rf.fit(train_features, train_labels)	Train mô hình random forest với dữ liệu train_features và train_labels
predictions = rf.predict(test_features)	Sử dụng phương pháp dự đoán của rừng trên dữ liệu thử nghiệm
print(confusion_matrix(test_labels,predictions.round()))	In ra confusion_matrix
print(classification_report(test_labels,predictions.round()))	In ra classification_report
print(accuracy_score(test_labels, predictions.round()))	In ra accuracy_score

```
confusion_matrix
[[ 36   2]
 [  1 104]]
classification_report
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        38
           1       0.98      0.99      0.99       105

   micro avg       0.98      0.98      0.98       143
   macro avg       0.98      0.97      0.97       143
weighted avg       0.98      0.98      0.98       143

accuracy_score
0.9790209790209791
```

Hình 16 : Hiển thị kết quả matriax, report và accuracy

2.2.2.6.Lập hàm tính accuracy theo công thức

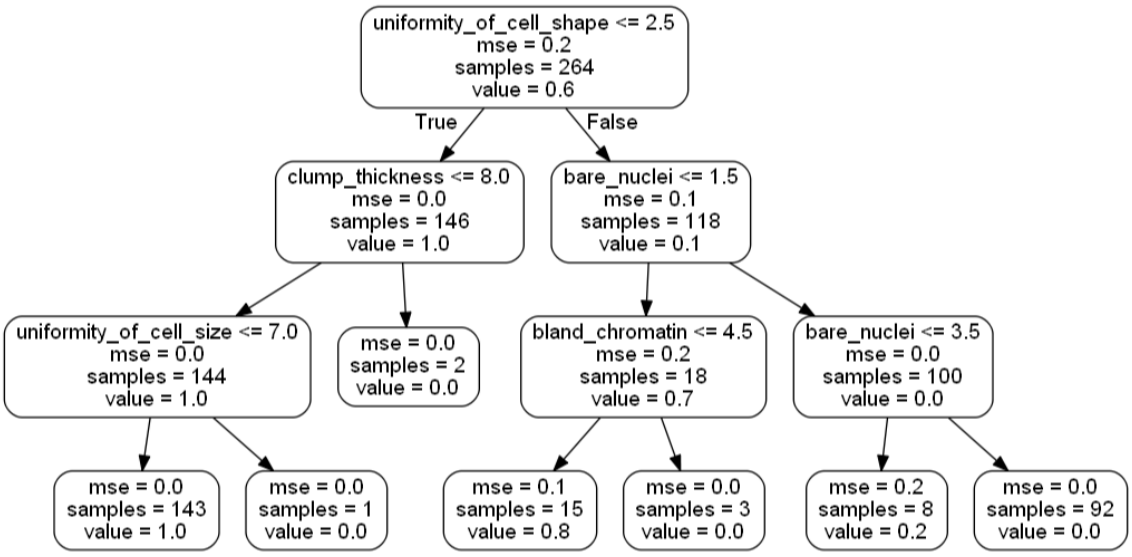
```
errors = abs(np.mean(test_labels) - np.mean(predictions))
print('Mean Absolute Error:', round(np.mean(errors),8), 'degrees.')
mape = 100*(np.mean(errors)/np.mean(test_labels))
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

errors = abs(predictions - test_labels)	Tính sai số tuyệt đối = trị tuyệt đối(predictions- tổng số target test)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')	In ra lỗi tuyệt đối trung bình
mape = 100*(errors/test_labels)	Tính phần trăm sai số tuyệt đối trung bình (MAPE)=100* (tổng số erro /tổng số target test)
accuracy = 100 - np.mean(mape)	Tính toán độ chính xác (Accuracy)=100- phần trăm sai số tuyệt đối trung bình (MAPE)
print('Accuracy:', round(accuracy, 2), '%.')	Hiển thị độ chính xác

2.2.2.7. Hiển thị một cây quyết định trong RandomForest

```
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
tree = rf.estimators_[1]
export_graphviz(tree, out_file = 'tree.dot',\
                feature_names = feature_list, rounded = True, precision = 1)
(graph, ) = pydot.graph_from_dot_file('tree.dot')
graph.write_png('C:/Users/user/Downloads/tree.png')
```

Code	Ý nghĩa
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'	Đưa đường dẫn thư viện Graphviz2.38 vào trong PATH
tree = rf.estimators_[1]	Lấy ra cây ở vị trí số 1 trong rf
export_graphviz(tree, out_file = 'tree.dot',feature_names = feature_list, rounded = True, precision = 1)	Xuất hình ảnh của cây thành một tập tin 'tree.dot' với các thông số mặc định
(graph,) = pydot.graph_from_dot_file('tree.dot')	Xử dụng tập tin tree.dot để tạo biểu đồ
graph.write_png('C:/Users/user/Downloads/tree.png')	Viết biểu đồ lên file tree.png



Hình 17 : Hiển thị cây thứ 1 trong Random Forest vừa tạo

2.2.2.8. Độ quan trọng của từng feature

```
importances = list(rf.feature_importances_)
feature_importances = [(feature, round(importance, 2))\
            for feature, importance in zip(feature_list, importances)]
feature_importances = \
    sorted(feature_importances, key = lambda x: x[1], reverse = True)
for pair in feature_importances :
    print('Variable: {:20} Importance: {}'.format(*pair));
```

Code	Ý nghĩa
importances = list(rf.feature_importances_)	Lấy độ quan trọng của feature
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature_list, importances)]	Danh sách các bộ dữ liệu với biến và tầm quan trọng

feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)	Sắp xếp các tính năng quan trọng theo quan trọng nhất trước tiên
for pair in feature_importances : print('Variable: {:20} Importance: {}'.format(*pair));	In ra các tính năng và quan trọng

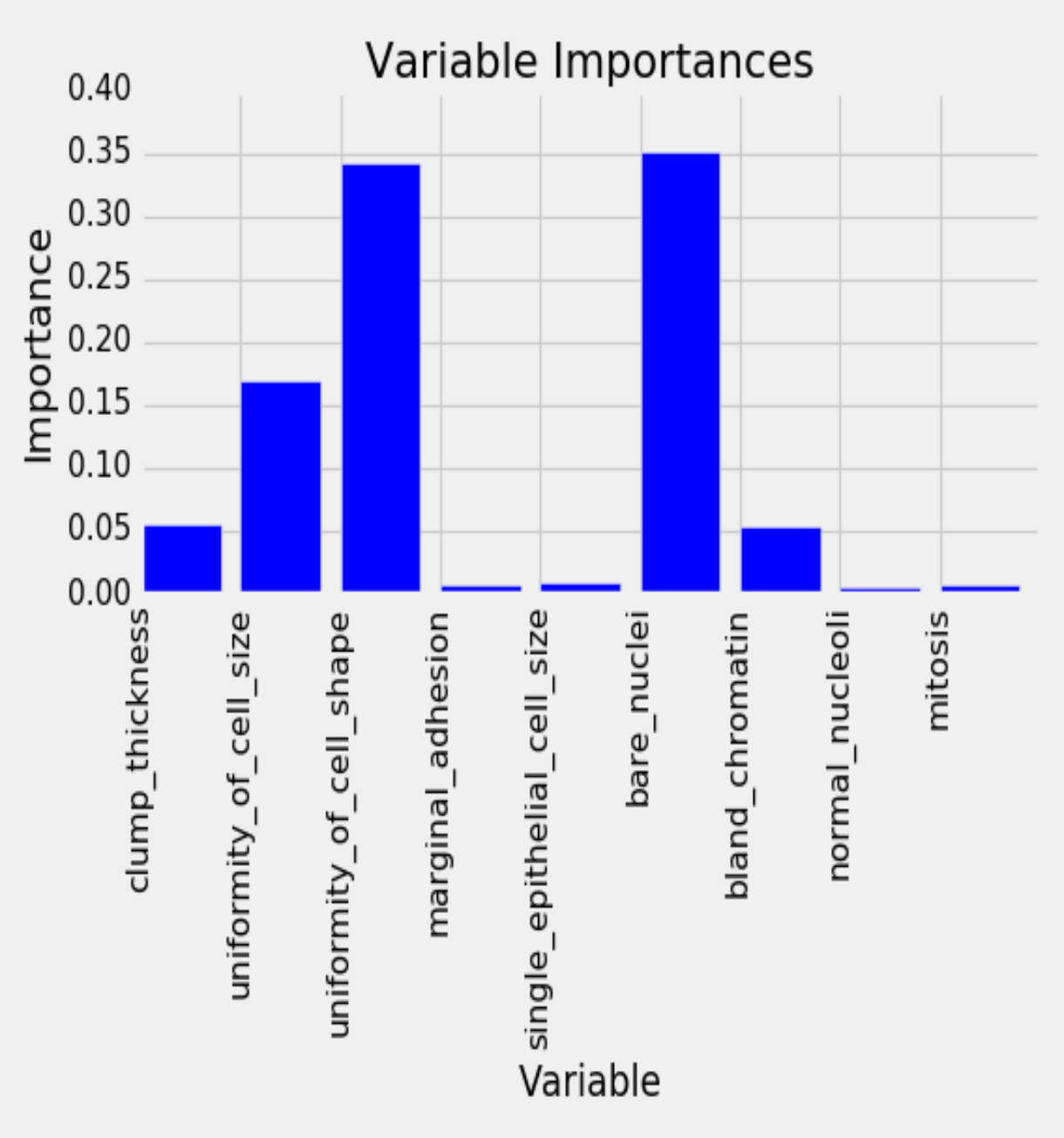
```
Variable: bare_nuclei          Importance: 0.35
Variable: uniformity_of_cell_shape Importance: 0.34
Variable: uniformity_of_cell_size Importance: 0.17
Variable: clump_thickness      Importance: 0.06
Variable: bland_chromatin      Importance: 0.05
Variable: marginal_adhesion    Importance: 0.01
Variable: single_epithelial_cell_size Importance: 0.01
Variable: mitosis              Importance: 0.01
Variable: normal_nucleoli      Importance: 0.0
```

Hình 18 : Hiển thị Độ quan trọng của từng feature

2.2.2.9. Biểu đồ trực quan hóa độ quan trọng của từng feature

```
plt.style.use('fivethirtyeight')
x_values = list(range(len(importances)))
plt.bar(x_values, importances, orientation = 'vertical')
plt.xticks(x_values, feature_list, rotation='vertical')
plt.ylabel('Importance'); plt.xlabel('Variable');
plt.title('Variable Importances');
plt.show()
```

Code	Ý nghĩa
plt.style.use('fivethirtyeight')	Đặt kiểu 'fivethirtyeight' cho hình biểu đồ
x_values = list(range(len(importances)))	Danh sách các tọa độ x cho biểu đồ
plt.bar(x_values, importances, orientation = 'vertical')	Lập biểu đồ cột
plt.xticks(x_values, feature_list, rotation='vertical')	Đánh nhãn cho trục x
plt.ylabel('Importance');	Nhãn trục y
plt.xlabel('Variable');	Nhãn trục x
plt.title('Variable Importances');	Tiêu đề của biểu đồ
plt.show()	Hiển thị biểu đồ



Hình 19 : Hiển thị Biểu đồ trực quan hóa độ quan trọng của từng feature

2.2.3. Test thử việc thay đổi theo giá trị accuracy_score

Train 75% - Test 25%		Số lượng DecisionTree			
		1	10	20	30
Độ sâu tối đa của mỗi DecisionTree	3	0.958	0.979	0.979	0.979
	5	0.951	0.951	0.972	0.972
	Cực đại	0.923	0.965	0.979	0.965

Train 50% - Test 50%		Số lượng DecisionTree			
		1	10	20	30
Độ sâu tối đa của mỗi DecisionTree	3	0.912	0.954	0.951	0.954
	5	0.933	0.951	0.947	0.954
	Cực đại	0.919	0.944	0.937	0.954

Train 25% - Test 75%		Số lượng DecisionTree			
		1	10	20	30
Độ sâu tối đa của mỗi DecisionTree	3	0.883	0. 941	0.944	0.956
	5	0.897	0.941	0.941	0.953
	Cực đại	0.897	0.941	0.941	0.953

2.3. Đánh giá

- Sau 3 lần thiết lập tỉ lệ training và testing data thì cả 3 đều có kết quả cao mặc dù khi giảm tỉ lệ dữ liệu tranning thì kết quả có giảm đi nhưng không đáng kể.

- Tuy nhiên thuật toán sẽ chạy chậm nếu như tạo ra số lượng decision tree lớn và random forest chỉ phù hợp với những dataset có các feature chứa đựng đầy đủ thông tin

- Random forest có khả năng tìm ra thuộc tính quan trọng nhất trong các thuộc tính của dataset và có thể chỉ ra những thuộc tính có tính quan trọng thấp hơn được biểu diễn bằng đồ thị