ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐH KHOA HỌC TỰ NHIÊN

★ ★ ★

BÁO CÁO CUỐI KÌ

ĐỀ TÀI: PHÂN LOẠI NHO KHÔ BẰNG THUẬT TOÁN K-NEAREST NEIGHBOR

GV: HÀ VĂN THẢO

MÔN HỌC: NHẬP MÔN KHOA HỌC DỮ LIÊU

LỚP: 20KDL-TTH

SVTH: 20280015 PHÙ CHÍ ĐẠT

20280018 DƯƠNG VI DOANH

20280024 TRẦN NGUYỄN NHẤT DUY

20280029 NGUYỄN NHẬT HÀO

TP HCM, Ngày 25 tháng 6 năm 2022

MỤC LỤC

I. Giới thiệu giải thuật KNN (K-Nearest Neighbors)..... 3

1. Tổng quan 3

2. Các phần quan trọng trong KNN 4

II. Giới thiệu và phân tích bộ dữ liệu..... 10

1. Giới thiệu 10

2. Phân tích.....12

III. Tiền xử lý dữ liệu và train model..... 18

IV. Kết luận..... 23

I. Giới thiệu giải thuật KNN (K-Nearest Neighbors)

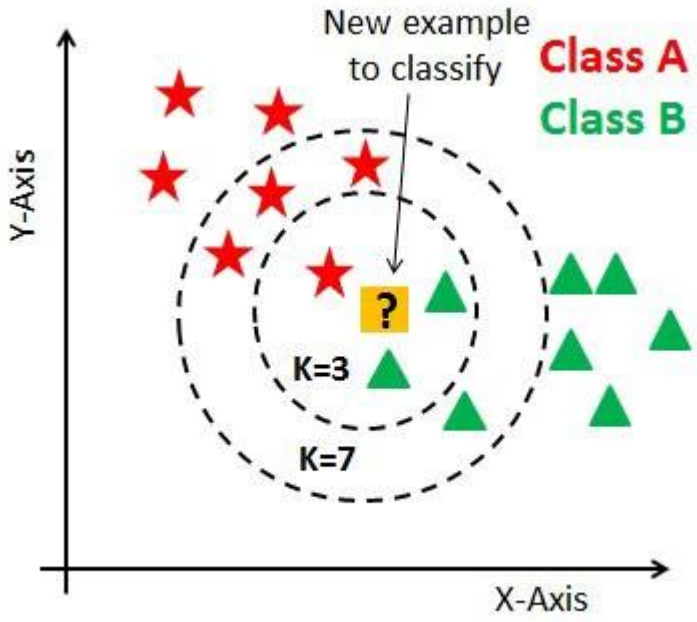
1. Tổng quan

K-Nearest Neighbor (hay K-lân cận) là một trong những thuật toán supervised-learning đơn giản trong Machine Learning. Khi training, thuật toán này không thật sự học từ dữ liệu training, mọi tính toán được thực hiện khi nó cần dự đoán kết quả cho dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán supervised-learning là Classification và Regression.

Trong bài toán Classification, label của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất rồi suy ra label.

Trong bài toán Regresssion, đầu ra của một điểm dữ liệu sẽ bằng trung bình trọng số của đầu ra những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó.

VD:



Trong trường hợp k = 3, điểm dữ liệu mới (ô vuông) nằm gần 2 tam giác và 1 ngôi sao, theo major voting thì ta sẽ dự đoán label của điểm dữ liệu mới thuộc class B (tam giác).

Trong trường hợp k = 7, điểm dữ liệu mới (ô vuông) nằm gần 3 tam giác và 4 ngôi sao, theo major voting thì ta sẽ dự đoán label của điểm dữ liệu mới thuộc class A (ngôi sao).

2. Các phần quan trọng trong KNN

❖ Chọn K

Đây là bước đầu tiên và có vai trò quan trọng trong thuật toán KNN, việc chọn K có quyết định tương đối lớn tới kết quả của thuật toán. Không có một công thức cụ thể nào cho việc chọn K, tuy nhiên có một số quy tắc thường gặp là:

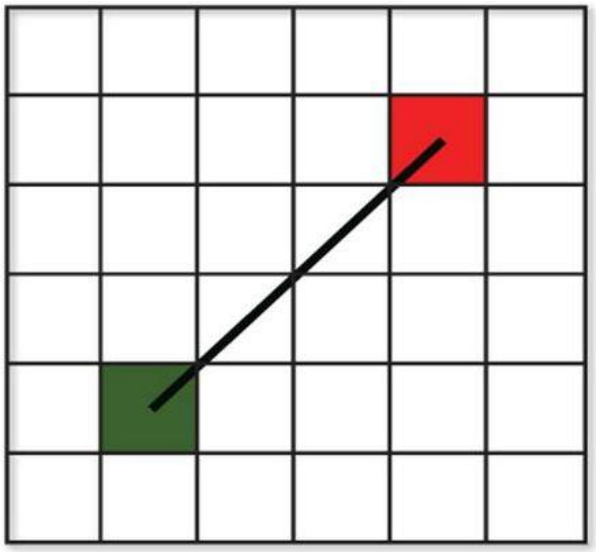
- Chọn K là số lẻ

- K = sqrt(n) với n là số dòng của data set
- Thực hiện tính toán trên nhiều K khác nhau, sau đó so sánh để chọn ra số K có kết quả tối ưu nhất, tuy nhiên việc này tương đối mất thời gian và độ hiệu quả mang lại không thật sự quá nhiều.

❖ Chọn công thức tính khoảng cách

KNN là thuật toán tính toán dựa trên khoảng cách để cho ra kết quả. Vì vậy việc chọn công thức tính khoảng cách có vai trò quan trọng và phụ thuộc vào tính chất của bộ data set.

Euclidean Distance

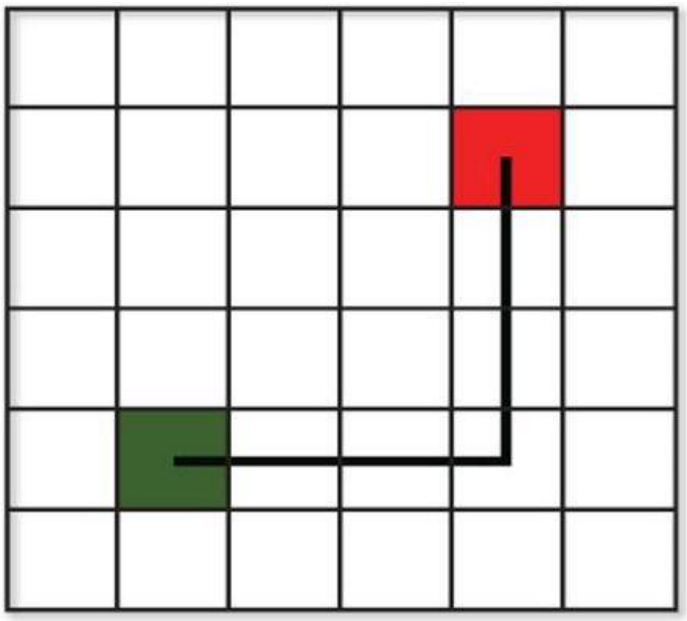


Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Đây là công thức cơ bản, được sử dụng nhiều nhất và là công thức mặc định trong các thư viện KNN.

Manhattan Distance



Manhattan Distance

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

Trong trường hợp bộ dữ liệu có số chiều (dimension) cao thì công thức này được sử dụng như là một phương án thay thế cho công thức Euclid, vì nó ít phức tạp hơn, làm cho tốc độ tính toán nhanh hơn.

Cosine Distance

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

Công thức này chủ yếu dùng để tính toán độ giống nhau giữa hai vector. Trong phân tích văn bản, nó thường được sử dụng để tính toán độ giống nhau giữa các tài liệu, hoặc tính toán số lần xuất hiện của một cụm từ trong một số tài liệu.

Công thức này cho ra các kết quả từ 0 tới 1, với giá trị 0 có nghĩa là các vector giống nhau 100% và 1 có nghĩa là chúng hoàn toàn khác nhau, không có điểm chung nào.

Hamming Distance

Thường đuọc sử dụng để so sánh hai chuỗi dữ liệu nhị phân. Khi so sánh hai chuỗi nhị phân có độ dài bằng nhau, khoảng cách Hamming là số bit khác nhau của 2 chuỗi. Phương pháp này duyệt qua toàn bộ dữ liệu và tìm các điểm dữ liệu giống nhau và khác nhau. Khoảng cách Hamming cho ra kết quả có bao nhiêu thuộc tính khác nhau.

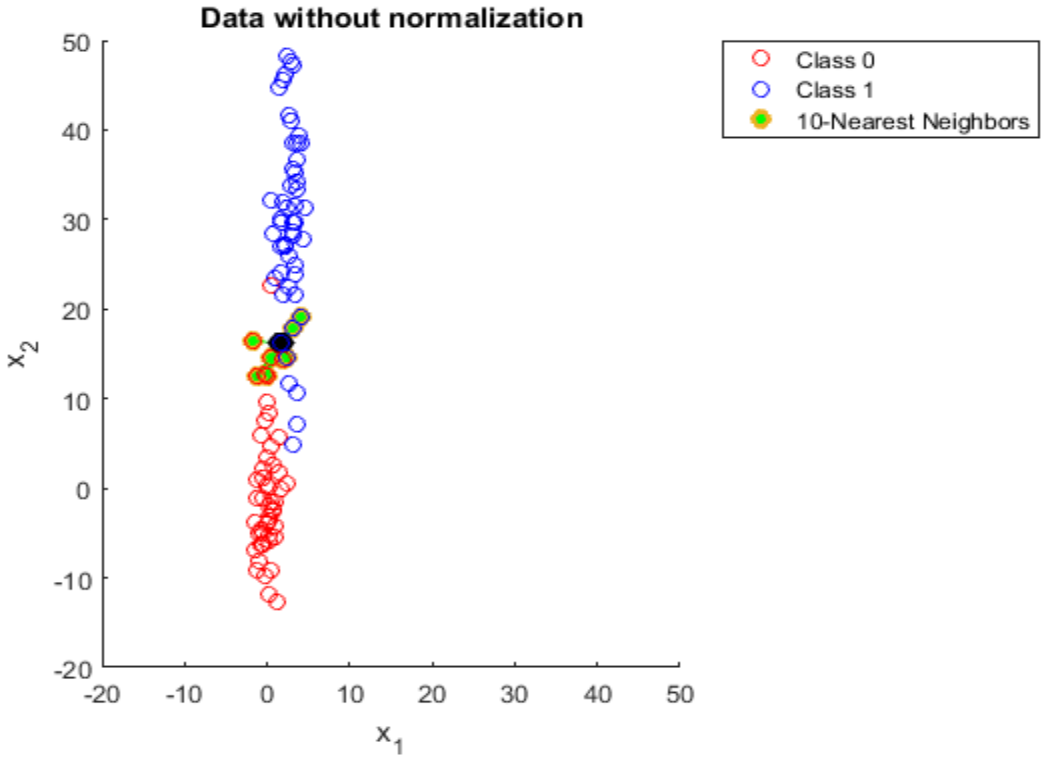
- Khoảng cách Hamming giữa 1011101 và 1001001 là 2.
- Khoảng cách Hamming giữa 2143896 và 2233796 là 3.
- Khoảng cách Hamming giữa "toned" và "roses" là 3.

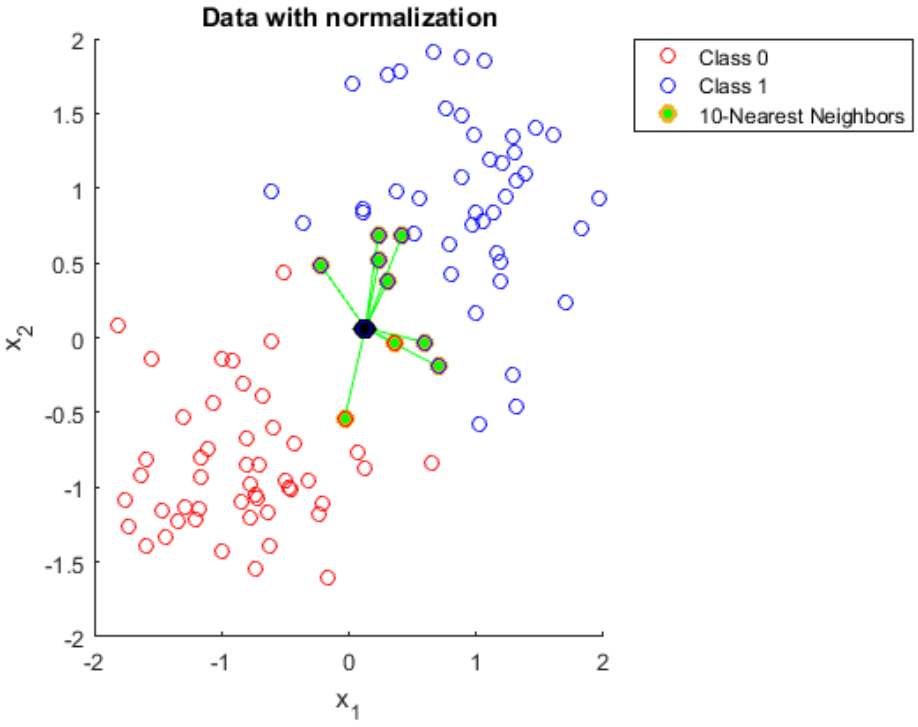
❖ Chuẩn hóa dữ liệu (Normalization)

Đây có thể xem là bước quan trọng nhất khi sử dụng KNN, nó có tác động rất lớn đến kết quả của thuật toán. Thông thường, các feature của một bộ data set sẽ có các khoảng giá trị khác nhau. Nếu như không chuẩn hóa, các feature có giá trị lớn sẽ chi phối, làm ảnh hưởng xấu đến kết quả khi tính khoảng cách. Ví dụ feature A có giá trị từ 0 - 1 và feature B có giá trị từ -100000 - 100000, nếu ta áp dụng công thức tính Euclid hay Manhattan để

tính khoảng cách thì kết quả sẽ hoàn toàn phụ thuộc vào giá trị của feature B, lúc này feature A trở nên vô nghĩa.

Ví dụ:





Thông thường khi chuẩn hóa, người ta sẽ đưa khoảng cách giá trị của các feature về mức 0-1 để thuận lợi cho việc tính toán và không làm ảnh hưởng xấu đến kết quả đầu ra.

Công thức chuẩn hóa thường dùng:

X_std = (X - X.min) / (X.max - X.min)

X_scaled = X_std * (max - min) + min

Với (max – min) là khoảng cách giá trị của feature đó.

❖ Ưu – nhược điểm

Ưu điểm:

- Thuật toán đơn giản, dễ hiểu, dễ triển khai.
- Độ phức tạp tính toán của quá trình training bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Sử dụng được trong phần lớn các bài toán phân loại.

Nhược điểm:

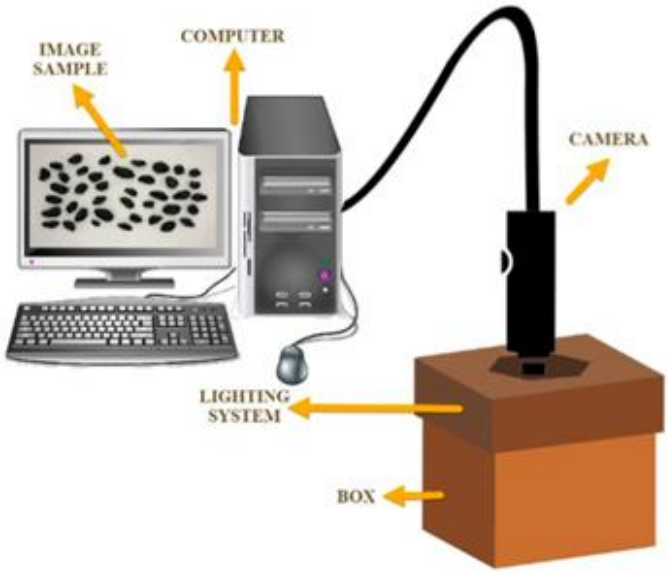
- Với K nhỏ sẽ dễ gặp các dữ liệu nhiễu, dẫn tới kết quả sai lệch.
- Nếu data set có số chiều lớn hoặc quá nhiều điểm dữ liệu thì tốn nhiều thời gian tính toán khoảng cách với từng điểm dữ liệu trong bộ training set.
- Cần biến đổi, chuẩn hóa data set trước khi sử dụng.

II. Giới thiệu và phân tích bộ dữ liệu

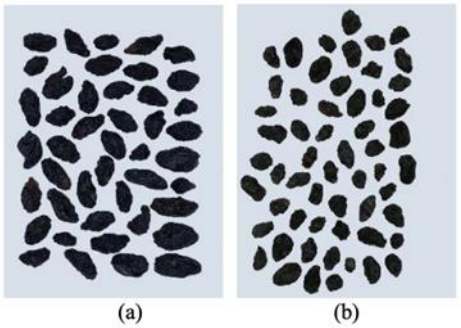
1. Giới thiệu

Hình ảnh mẫu nho khô được thu thập và xử lý bằng cách sử dụng các kỹ thuật xử lý ảnh khác nhau.

Hình ảnh nho khô được chụp bằng cách sử dụng một hộp kín bốn cạnh có gắn camera và cơ chế chiếu sáng bên trong. Lý do hộp được bao bọc là để ngăn nó nhận được ánh sáng ngược từ bên ngoài và ngăn chặn sự hình thành bóng trên các mẫu nho khô. Sàn hộp được đặt thành màu trắng để có thể dễ dàng phân biệt nho khô trong quá trình xử lý hình ảnh.

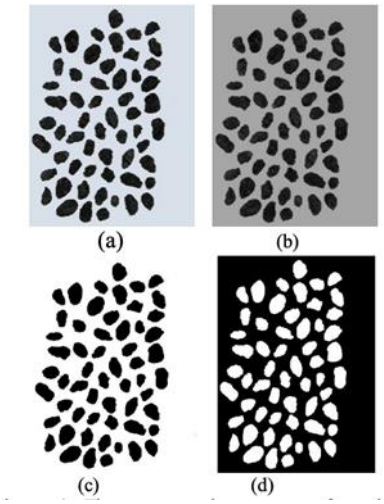


Hệ thống đã thu được 900 hình ảnh, với 450 thuộc loại Besni và 450 thuộc loại Kecimen.



Hình ảnh các mẫu của nhiều loại nho khô (a. Besni, b. Kecimen)

Việc xử lý hình ảnh được thực hiện bằng ứng dụng MATLAB. Hình ảnh thu được từ máy ảnh được chuyển đổi chủ yếu sang thang độ xám và sau đó là hình ảnh nhị phân để chuẩn bị cho giai đoạn trích xuất tính năng. Vì màu nho khô gần với màu đen hơn nên tông màu nền được chọn là màu trắng. Bằng cách sử dụng hàm imcomplement trên ảnh nhị phân, các vùng trắng được chuyển đổi thành đen và các vùng đen thành trắng. Với quá trình trên, hình ảnh cuối cùng sẽ không bị nhiễu.



((a) ảnh thực, (b) ảnh thang độ xám, (c) ảnh nhị phân, (d) ảnh không bị nhiễu)

Sau quá trình xử lý, người ta rút ra được 7 feature như sau:

- Area (diện tích): Cung cấp số lượng pixel trong ranh giới của hạt nho khô.
- Perimeter (chu vi): đo bằng cách tính khoảng cách giữa ranh giới của hạt nho khô và các pixel xung quanh nó.
- MajorAxisLength (chiều dài trục chính): là đường dài nhất có thể vẽ được trên hạt nho khô.
- MinorAxisLength (chiều dài trục nhỏ): là đường ngắn nhất có thể vẽ được trên hạt nho khô.
- Eccentricity (độ lệch tâm): tỉ lệ khoảng cách giữa chiều dài tiêu cự và chiều dài trục chính (giá trị: 0~1, nếu độ lệch tâm = 0 thì hình ellipse là hình tròn)
- Convex Area: số lượng pixel nhỏ nhất phần lồi lên của vỏ hạt nho khô.
- Extent: tỷ lệ của vùng được hình thành bởi hạt nho khô trên tổng số pixel trong hộp giới hạn.

2. Phân tích

```
[ ] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Area             900 non-null    int64
 1   MajorAxisLength  900 non-null    float64
 2   MinorAxisLength  900 non-null    float64
 3   Eccentricity     900 non-null    float64
 4   ConvexArea       900 non-null    int64
 5   Extent           900 non-null    float64
 6   Perimeter        900 non-null    float64
 7   Class            900 non-null    object
dtypes: float64(5), int64(2), object(1)
memory usage: 56.4+ KB
```

Bộ dữ liệu đầy đủ, không chứa bất kỳ giá trị NULL nào.

```
data.Class.value_counts()

Kecimen    450
Besni      450
Name: Class, dtype: int64
```

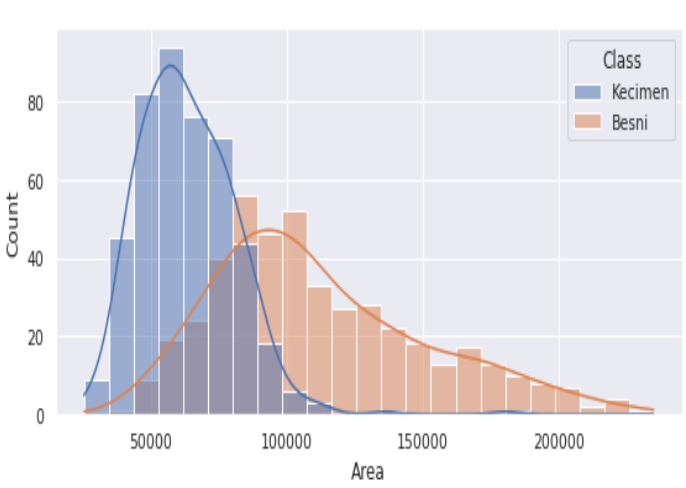
Class có 450 dòng dữ liệu thuộc loại nho Kecimen và 450 dòng thuộc loại nho Besni

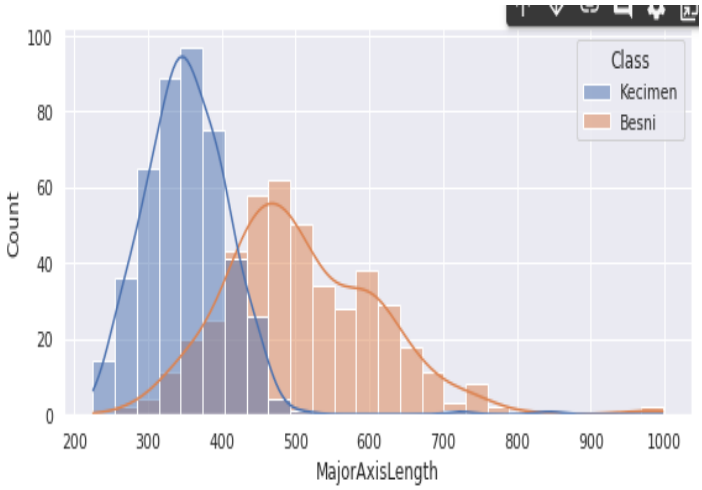
[45] data.head()

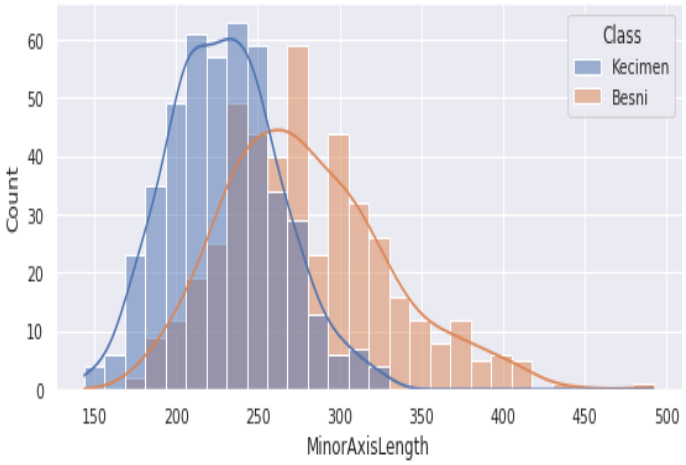
	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter	Class
0	87524	442.246011	253.291155	0.819738	90546	0.758651	1184.040	Kecimen
1	75166	406.690687	243.032436	0.801805	78789	0.684130	1121.786	Kecimen
2	90856	442.267048	266.328318	0.798354	93717	0.637613	1208.575	Kecimen
3	45928	286.540559	208.760042	0.684989	47336	0.699599	844.162	Kecimen
4	79408	352.190770	290.827533	0.564011	81463	0.792772	1073.251	Kecimen

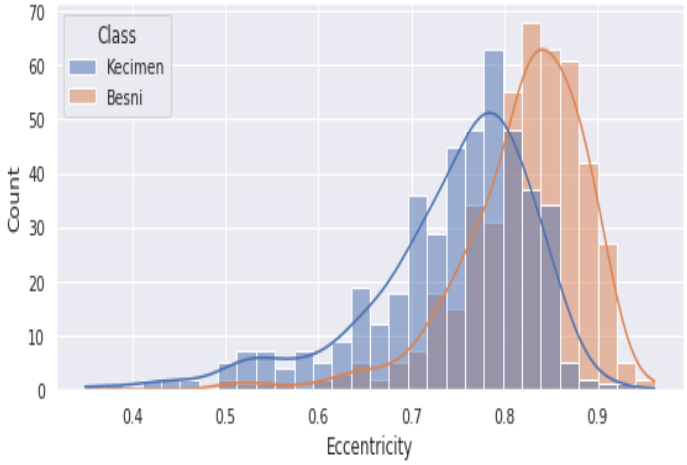
Giá trị của các feature trong bộ dữ liệu.

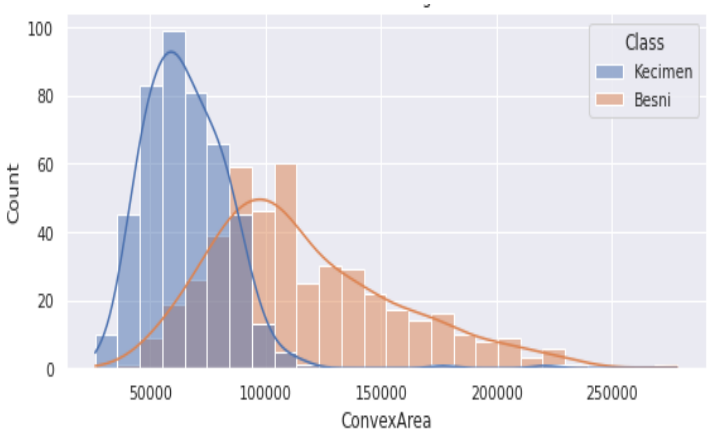
Đồ thị histogram và boxplot của các feature, phân loại theo class.

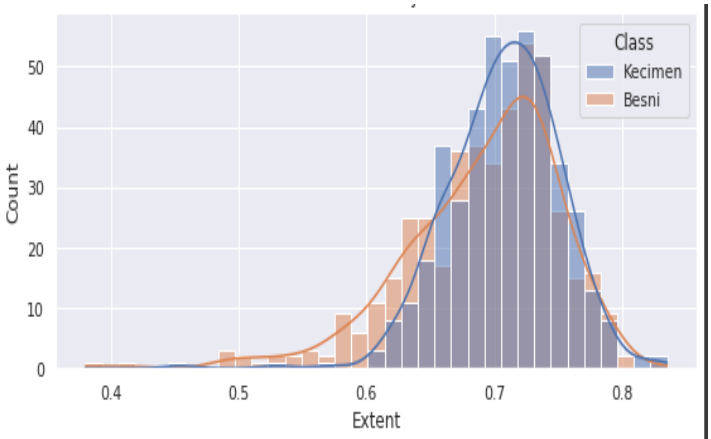


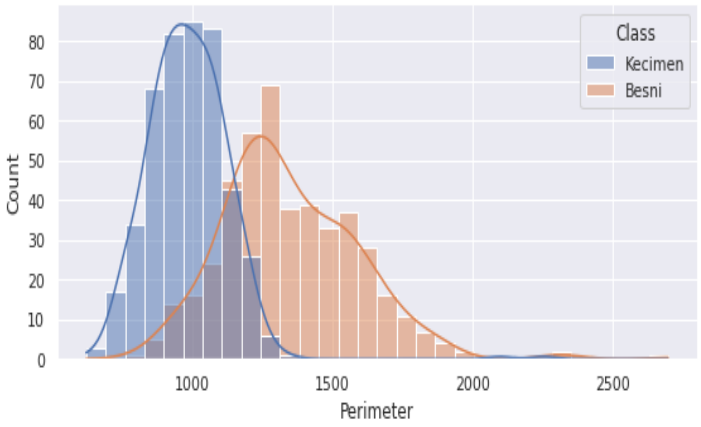


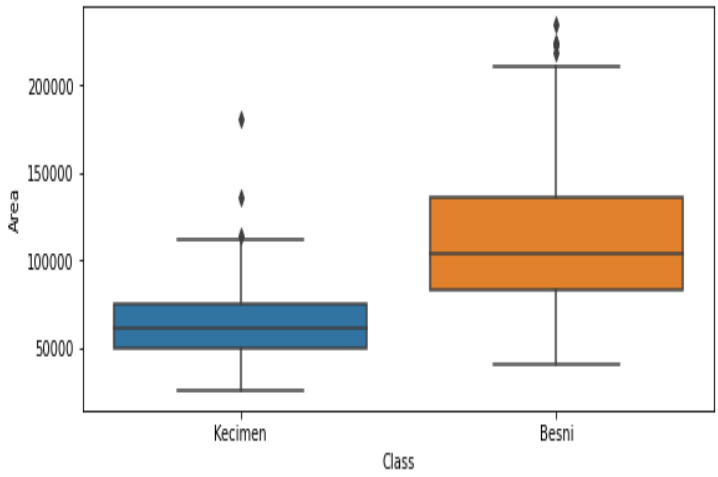


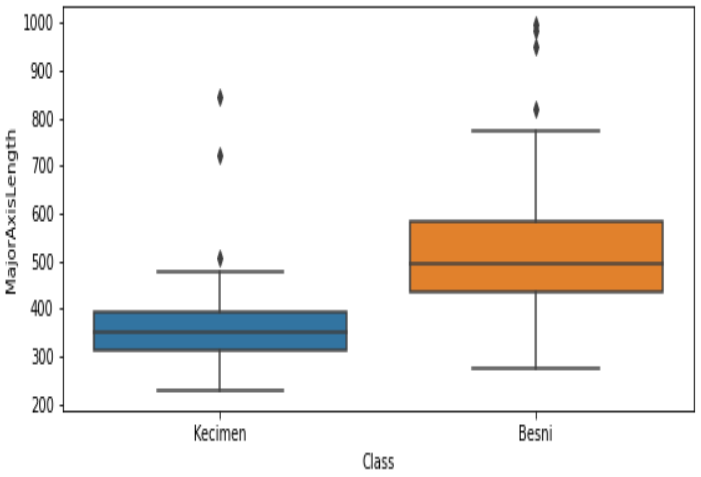


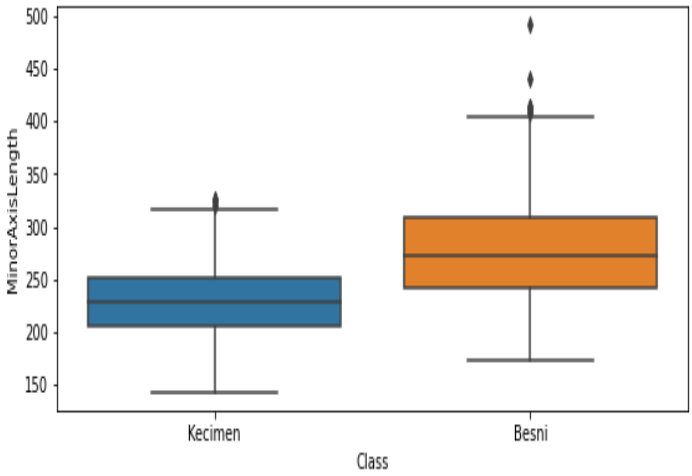

Class
Kecimen
Besni
Count
ConvexArea

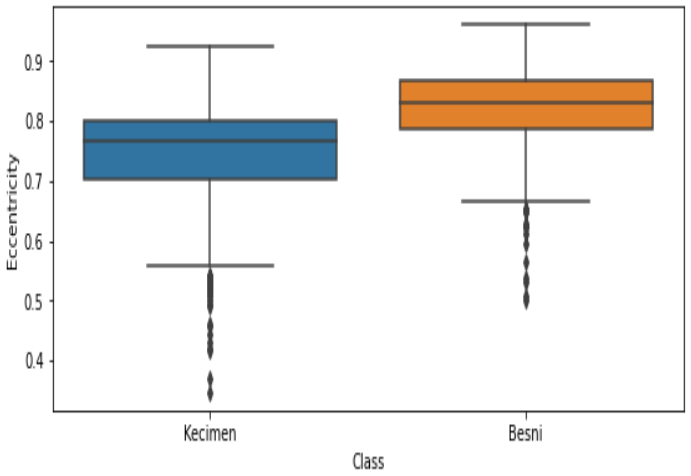

Class
Kecimen
Besni
Count
Extent

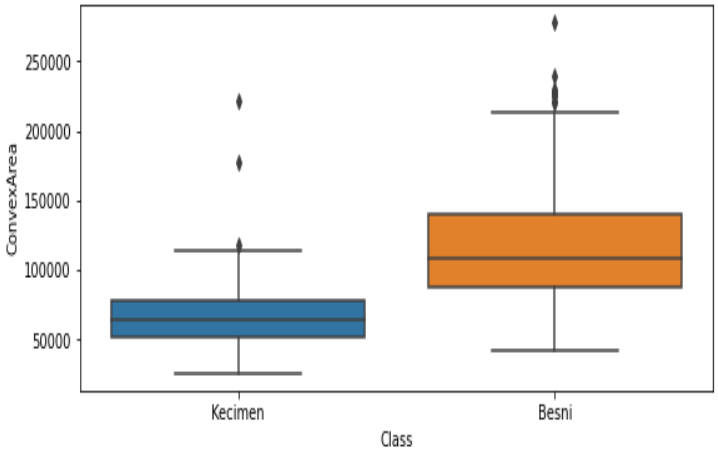

Class
Kecimen
Besni
Count
Perimeter

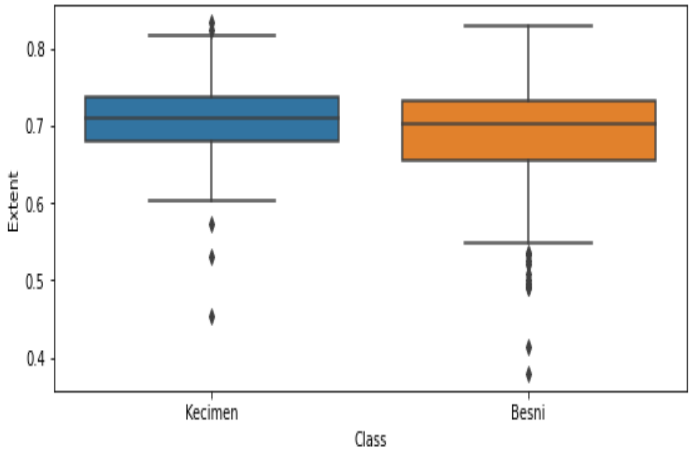

Area
Kecimen
Besni
Class

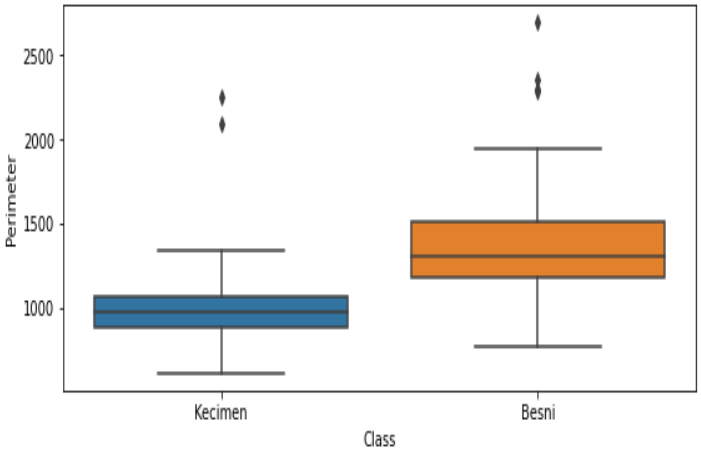

MajorAxisLength
Kecimen
Besni
Class


MinorAxisLength
Kecimen
Besni
Class


Eccentricity
Kecimen
Besni
Class





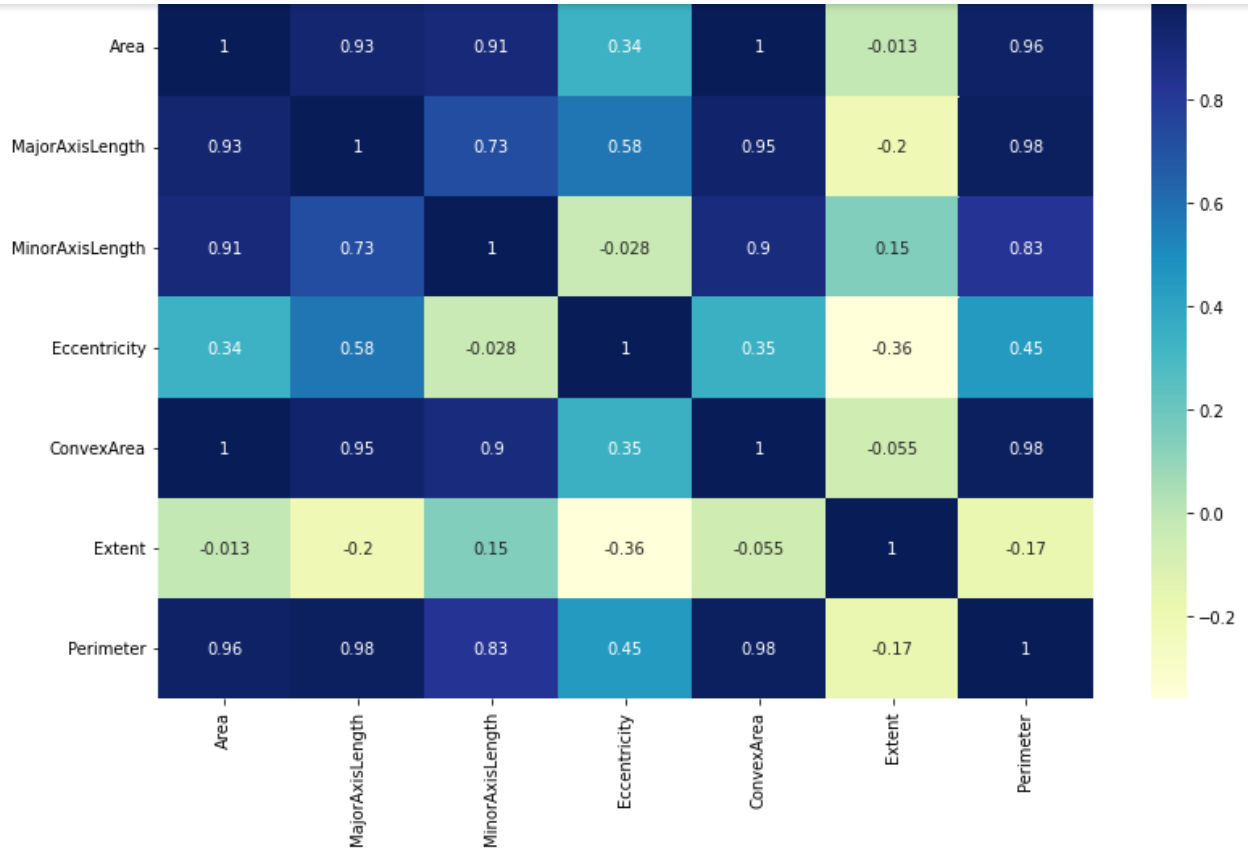


Từ đồ thị ta thấy nho Besni hầu như có các giá trị Area, Perimeter, MajorAxisLength, MinorAxisLength lớn hơn nho Kecimen, tuy nhiên độ phân tán các giá trị của nho Besni tương đối lớn, ngược lại các giá trị của nho Kecimen có mức phân tán thấp.

Ngoài ra, ta còn nhận thấy các giá trị MinorAxisLength, Eccentricity và Extent của 2 class có sự tương đồng tương đối lớn, đặc biệt là sự phân bố giá trị Extent của 2 class gần như không mấy khác biệt.

Ngược lại các giá trị Area, MajorAxisLength, ConvexArea, Perimeter lại có sự khác biệt tương đối lớn giữa 2 class.

Heatmap của bộ dữ liệu



Ta nhận thấy các thuộc tính trên mỗi hạt nho như Area, MajorAxisLength, MinorAxisLength, ConvexArea và Perimeter đều tương đồng với nhau, tức là một hạt nho có kích thước to thì độ dài trục chính, trục nhỏ và chu vi có giá trị lớn. Điều này cho thấy bộ dữ liệu đã được xây dựng tốt, hợp lý và đáng tin cậy.

III. Tiền xử lý dữ liệu và train model

```
[3] data['Class'] = OrdinalEncoder().fit_transform(data[['Class']])
    x = data.drop("Class", axis = 1)
    y = data.Class
```

```
y
```

```
0      1.0
1      1.0
2      1.0
3      1.0
4      1.0
      ...
895    0.0
896    0.0
897    0.0
898    0.0
899    0.0
Name: Class, Length: 900, dtype: float64
```

Chuyển hóa class về dạng 0, 1. Đồng thời gán class cho biến y, gán phần còn lại của data set cho x để phục vụ training, dự đoán kết quả.

Class Kecimen được chuyển hóa thành 1, Besni thành 0.

```
[11] cal_mean(x, y, 5000)

     Accuracy: 82.04 %
     Precision: 79.08 %
     Recall: 87.2 %
     F1 Score: 82.84 %
```

Thực hiện chạy KNN 5000 lần, K = 29 (bộ dữ liệu có 900 dòng, áp dụng K= sqrt(900) = 30, tuy nhiên ta nên chọn K là số lẻ nên chọn bằng 29) và tính kết quả trung bình. Dữ liệu lần chạy này chưa thực hiện normalize data.

Ta có Accuracy: 82.04%, Precision: 79.08%, Recall: 87.2% và F1 Score: 82.84%

```
[13] for i in x.columns:
         x[i]=preprocessing.MinMaxScaler().fit_transform(x[[i]])

     x.head()
```

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter
0	0.296370	0.280714	0.314376	0.767872	0.255504	0.831422	0.271791
1	0.237427	0.234638	0.284945	0.738636	0.208864	0.667854	0.241842
2	0.312263	0.280741	0.351778	0.733009	0.268084	0.565754	0.283594
3	0.097973	0.078935	0.186620	0.548194	0.084089	0.701809	0.108284
4	0.257660	0.164011	0.422064	0.350968	0.219472	0.906315	0.218493

Áp dụng hàm MinMaxScaler, để thực hiện normalize data.

Các giá trị đều được đưa về khoảng 0-1.

Công thức hàm:

X_std = (X - X.min) / (X.max - X.min)

X_scaled = X_std * (max - min) + min

```
cal_mean(x,y,5000)

Accuracy: 86.24 %
Precision: 83.34 %
Recall: 90.64 %
F1 Score: 86.78 %
```

Thực hiện lại việc dự đoán, ta thấy độ chính xác tăng lên tương đối.

Accuracy: 86.24 %, tăng 4.2 %

Precision: 83.34 %, tăng 4.26 %

Recall: 90.64 %, tăng 3.44 %

F1 Score: 86.78 %, tăng 3.94 %

```
data_new = SelectKBest(chi2, k=4).fit_transform(x,y)
data_new = pd.DataFrame(data_new)
data_new.head()
```

	0	1	2	3
0	0.296370	0.280714	0.255504	0.271791
1	0.237427	0.234638	0.208864	0.241842
2	0.312263	0.280741	0.268084	0.283594
3	0.097973	0.078935	0.084089	0.108284
4	0.257660	0.164011	0.219472	0.218493

Dùng thư viện SelectKBest để chọn ra 4 feature có khả năng phân loại tốt nhất, ta nhận được các feature giống với phần phân tích dữ liệu ở mục 2.

Ta được các feature: Area, MajorAxisLength, ConvexArea, Perimeter.

```
x_train2 = x.loc[:,['Area','MajorAxisLength', 'ConvexArea', 'Perimeter' ]]
x_train2
```

	Area	MajorAxisLength	ConvexArea	Perimeter
0	0.296370	0.280714	0.255504	0.271791
1	0.237427	0.234638	0.208864	0.241842
2	0.312263	0.280741	0.268084	0.283594
3	0.097973	0.078935	0.084089	0.108284
4	0.257660	0.164011	0.219472	0.218493
...	...	...	...	...
895	0.275975	0.264945	0.236831	0.245347
896	0.295540	0.278757	0.256905	0.286325
897	0.354240	0.267056	0.317858	0.324126
898	0.324983	0.324902	0.283698	0.307635
899	0.287236	0.371214	0.250153	0.314521

900 rows × 4 columns

Thực hiện tách 4 feature từ data set.

```
cal_mean(x_train2, y, 5000)
```

```
Accuracy: 85.49 %
Precision: 82.75 %
Recall: 89.71 %
F1 Score: 86.02 %
```

Thực hiện dự đoán kết quả dựa trên 4 feature được tách khi nãy. Ta thấy kết quả có giảm nhưng không đáng kể

Accuracy: 85.49%, giảm 0.55 %

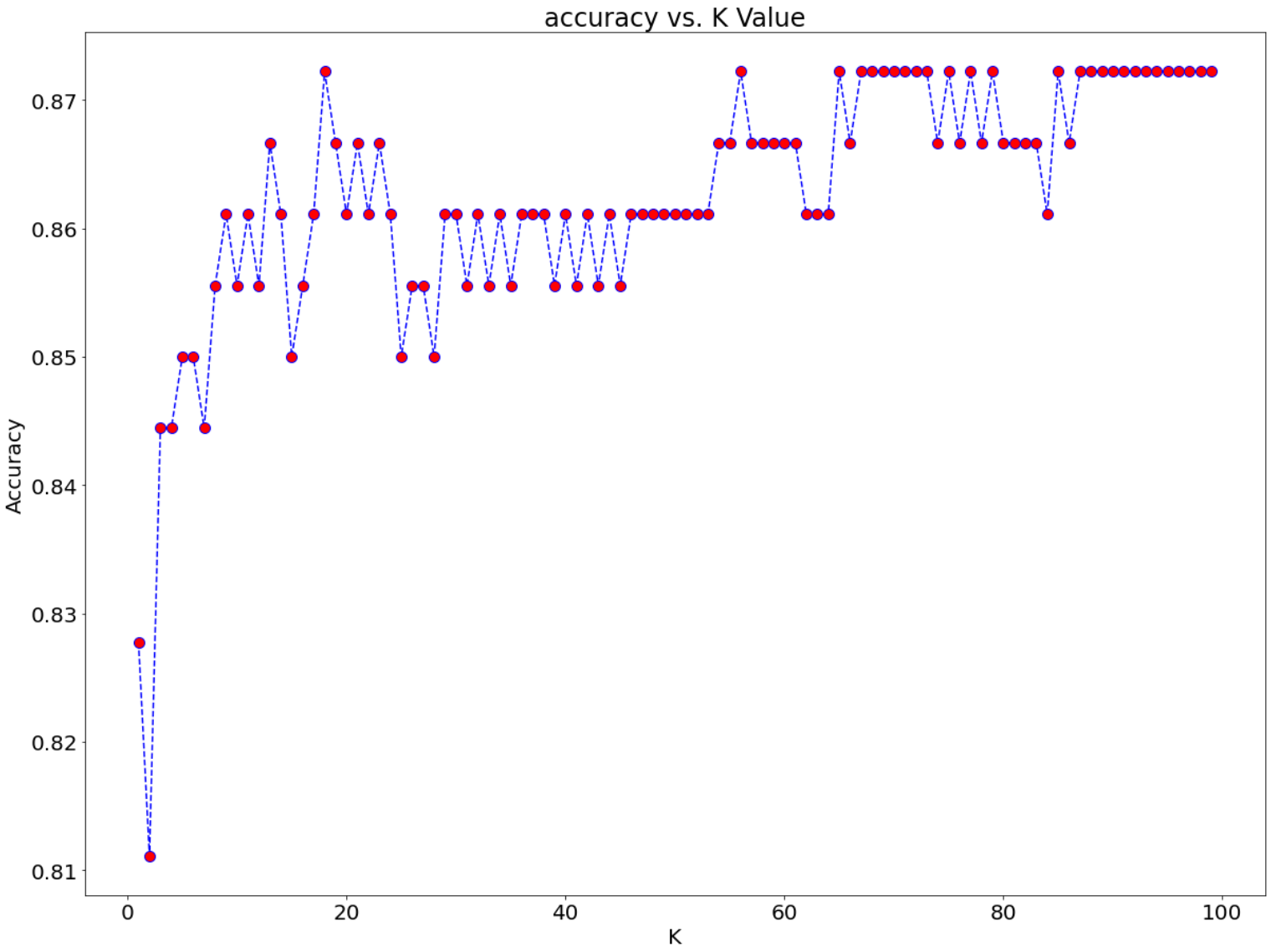
Precision: 82.75%, giảm 0.59 %

Recall: 89.71 %, giảm 0.93 %

F1 Score: 86.02 %, giảm 0.76 %

Thực hiện kiểm tra nhiều giá trị K khác nhau (đã normalized data):

```
Maximum accuracy:- 0.8722222222222222 at K = 17
```



Ta nhận thấy accuracy của các giá trị K xung quanh giá trị K = sqrt(n) = 30 giao động không quá nhiều, chỉ chênh lệch nhau

một vài %. Vậy ta chọn K vào đúng một khoảng K phù hợp tương ứng với bộ dữ liệu thì accuracy sẽ đạt tối ưu, sau đó ta nên chú trọng vào việc data cleanning để đạt được accuracy tốt hơn chứ không nên quá chú tâm vào việc chọn đúng một giá trị K chính xác. Ngoài ra, việc K quá nhỏ (<10) sẽ làm cho độ chính xác giảm đi tương đối đáng kể.

IV. Kết luận

Bài báo cáo đã giới thiệu sơ lược về thuật toán K-Nearest Neighbor và áp dụng vào bộ data set Raisin trên Kaggle.

Qua bài báo cáo, ta nhận thấy việc chuẩn hóa dữ liệu có vai trò quan trọng trong thuật toán KNN, nó giúp việc tính toán được dễ dàng và chính xác hơn.

Sau khi phân tích, ta tách ra được 4 feature có độ phân loại cao, điều này giúp cải thiện tốc độ tính toán nếu như bộ dữ liệu có nhiều điểm hoặc nhiều chiều mà không làm độ chính xác giảm đi đáng kể.

Reference:

https://www.kaggle.com/datasets/muratkokludataset/raisin-dataset

https://dergipark.org.tr/tr/download/article-file/1227592

https://machinelearningcoban.com/2017/01/08/knn

https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

Cảm ơn thầy và các bạn đã xem qua bài báo cáo.

Chúc mọi người một ngày vui vẻ.