

Knn ve NaiveBayes Sınıflandırılması ile İris Çiçeği Analizi

ÖZ

Bu projede, iris çiçeğinin K-NN (K-En Yakın Komşu) sınıflandırması, bir örneğin sınıf etiketini belirlemek için örnekleri karşılaştırır ve en yakın k komşuyu bulur. Bu komşuların etiketleri, örneklerin sınıfını tahmin etmek için kullanılır. Örneğin, bir iris çiçeğinin çiçek taç yaprak uzunluğu ve genişliği özelliklerine göre sınıflandırılması için, k-NN algoritması önce bu özellikleri kullanarak en yakın k komşuyu bulur ve bu komşuların sınıf etiketlerini kullanarak çiçeğin sınıfını tahmin eder. Naive Bayes sınıflandırması, olasılık teorisine dayanan bir sınıflandırma yöntemidir. Bu yöntem, özelliklerin sınıf etiketini tahmin etmek için ne kadar önemli olduğunu hesaplar. Örneğin, bir iris çiçeğinin çiçek taç yaprak uzunluğu ve genişliği özelliklerine göre sınıflandırılması için, Naive Bayes algoritması önce her özelliğin hangi sınıf için ne kadar önemli olduğunu hesaplar ve sonra bu bilgiyi kullanarak çiçeğin sınıfını tahmin eder. İris çiçeği analizi, sınıflandırma problemleri için sıkça kullanılan bir örnek olmasının yanı sıra, K-NN ve Naive Bayes sınıflandırma algoritmalarının karşılaştırılması için de sıklıkla kullanılan bir örnektir. Bu iki yöntem arasında, hangisinin daha iyi performans gösterdiği, veri setine ve özelliklere bağlı olarak değişebilir.

Giriş

İris adındaki bitkinin 3 farklı türüne ait (İris Setosa, İris Virginica, Iris Versicolor) her türden 50 örnek olmak koşuluyla 150 örneğe sahip bir veri kullanılarak K-NN ve Naive Bayes sınıflandırma algoritması kullanılmasını o

Her bir çiçeğe ait 4 özellik tanımlanmıştır;

Sepal-length(Alt-çanak yaprak uzunluğu)
Sepal with(Alt yaprak genişliği)
Pedal-length(Üst-taç yaprak uzunluğu)
Pedal-with(Üst yaprak genişliği)

Bu özelliklere bakılarak bitkinin 3 türden hangisine ait olduğu tahmin edilebilmektedir. Biz de bu 4 özelliğini kullanarak İris çiçeğinin hangi türe ait olduğunu makine öğrenmesi yoluyla tahmin etmeye çalışacağız.

Burada her bir türe ait tanımlanan özellikler(Alt-çanak yaprak uzunluğu, alt yaprak genişliği, üst-taç yaprak uzunluğu, üst yaprak genişliği) bağımsız değişkenler, tür isimleri (İris Setosa, İris Virginica, Iris Versicolor) ise bağımlı değişkenlerimiz olacaktır.

Yöntem

İris çiçeği bitkisinin 3 farklı türüne ait sınıflandırılması işlemi, K-NN ve Naive Bayes sınıflandırma algoritması analizi ile yapılmaktadır. Her bir çiçeğin bazı özelliklerinin (boy, genişlik vs.) yanı sıra her tür için 50 örnek içeren 3 iris türünü içerir.

Bulgular

Araştırmanın materyalini, İris çiçeği bitkisinin 3 farklı türüne ait (İris Setosa, İris Virginica, Iris Versicolor) her türden 50 örnek olmak koşuluyla 150 örneğe sahip bir veri seti oluşturmaktadır.

150 adet olan veri seti içerisinde kullanılan verilerden farklı türlere ait 5'er adet data aşağıdaki gibidir.

No.	sepalength	sepalwidth	petallength	petalwidth	class
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
51	7.0	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4.0	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
101	6.3	3.3	6.0	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3.0	5.9	2.1	Iris-virginica

Relation: iris						
No.	1: sepalength Numeric	2: sepalwidth Numeric	3: petallength Numeric	4: petalwidth Numeric	5: class Nominal	
1	5.1	3.5	1.4	0.2	Iris-setosa	
2	4.9	3.0	1.4	0.2	Iris-setosa	
3	4.7	3.2	1.3	0.2	Iris-setosa	
4	4.6	3.1	1.5	0.2	Iris-setosa	
5	5.0	3.6	1.4	0.2	Iris-setosa	
6	5.4	3.9	1.7	0.4	Iris-setosa	
7	4.6	3.4	1.4	0.3	Iris-setosa	
8	5.0	3.4	1.5	0.2	Iris-setosa	
9	4.4	2.9	1.4	0.2	Iris-setosa	
10	4.9	3.1	1.5	0.1	Iris-setosa	
11	5.4	3.7	1.5	0.2	Iris-setosa	

KNN (K-Nearest Neighbors) Algoritması

K-NN algoritması en basit ve en çok kullanılan sınıflandırma algoritmasından biridir. K-NN parametrik olmayan, lazy (tembel) bir öğrenme algoritmasıdır. Lazy kavramını anlamaya çalışırsak eager learning aksine lazy learning'in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini "ezberler". Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar.

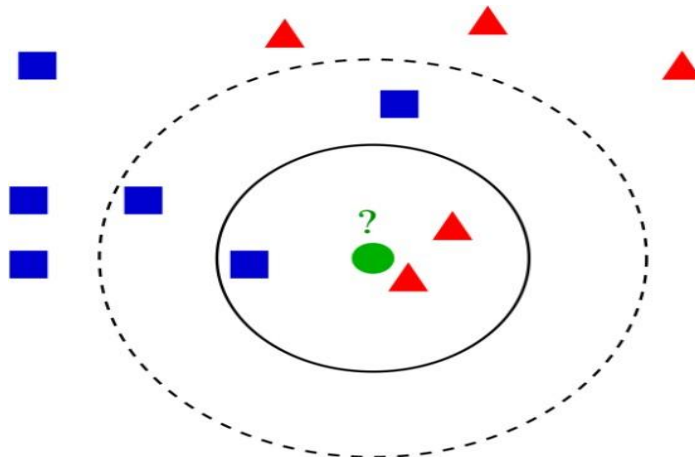
Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır. Öklid fonksiyonuna alternatif olarak Manhattan, Minkowski ve Hamming fonksiyonlarında kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır.

Uygulama yapılan veri setini analizi için bir çok algoritma algoritma bulunmaktadır. Bunların içinden KNN (K-Nearest Neighbors) Algoritmasından bir örnek ile bahsedeceğim.

KNN en basit anlamı ile içerisinde tahmin edilecek değerlerin bağımsız değişkenlerinin oluşturduğu vektörün en yakın komşularının hangi sınıfta yoğun olduğu bilgisi üzerinden sınıfını tahmin etmeye dayanır.

KNN (K-Nearest Neighbors) Algoritması iki temel değer üzerinden tahmin yaparak ilerleyeceğiz;

- **Distance (Uzaklık):** Tahmin edilecek noktanın diğer noktalara uzaklığı hesaplanır. Bunun için Minkowski uzaklık hesaplama fonksiyonu kullanılır.
- **K (komşuluk sayısı):** En yakın kaç komşu üzerinden hesaplama yapılacağını söyleriz. K değeri sonucu direkt etkileyecektir. K 1 olursa overfit etme olasılığı çok yüksek olacaktır. Çok büyük olursada çok genel sonuçlar verecektir. Bu sebeple optimum K değerini tahmin etmek analizin asıl konusu olarak karşımızda durmaktadır. K değerinin önemini aşağıdaki grafik çok güzel bir şekilde göstermektedir. Eğer K=3 (düz çizginin olduğu yer) seçersek sınıflandırma algoritması ? işareti ile gösterilen noktayı, kırmızı üçgen sınıfı olarak tanımlayacaktır. Fakat K=5 (kesikli çizginin olduğu alan) seçersek sınıflandırma algoritması, aynı noktayı mavi kare sınıfı olarak tanımlayacaktır.



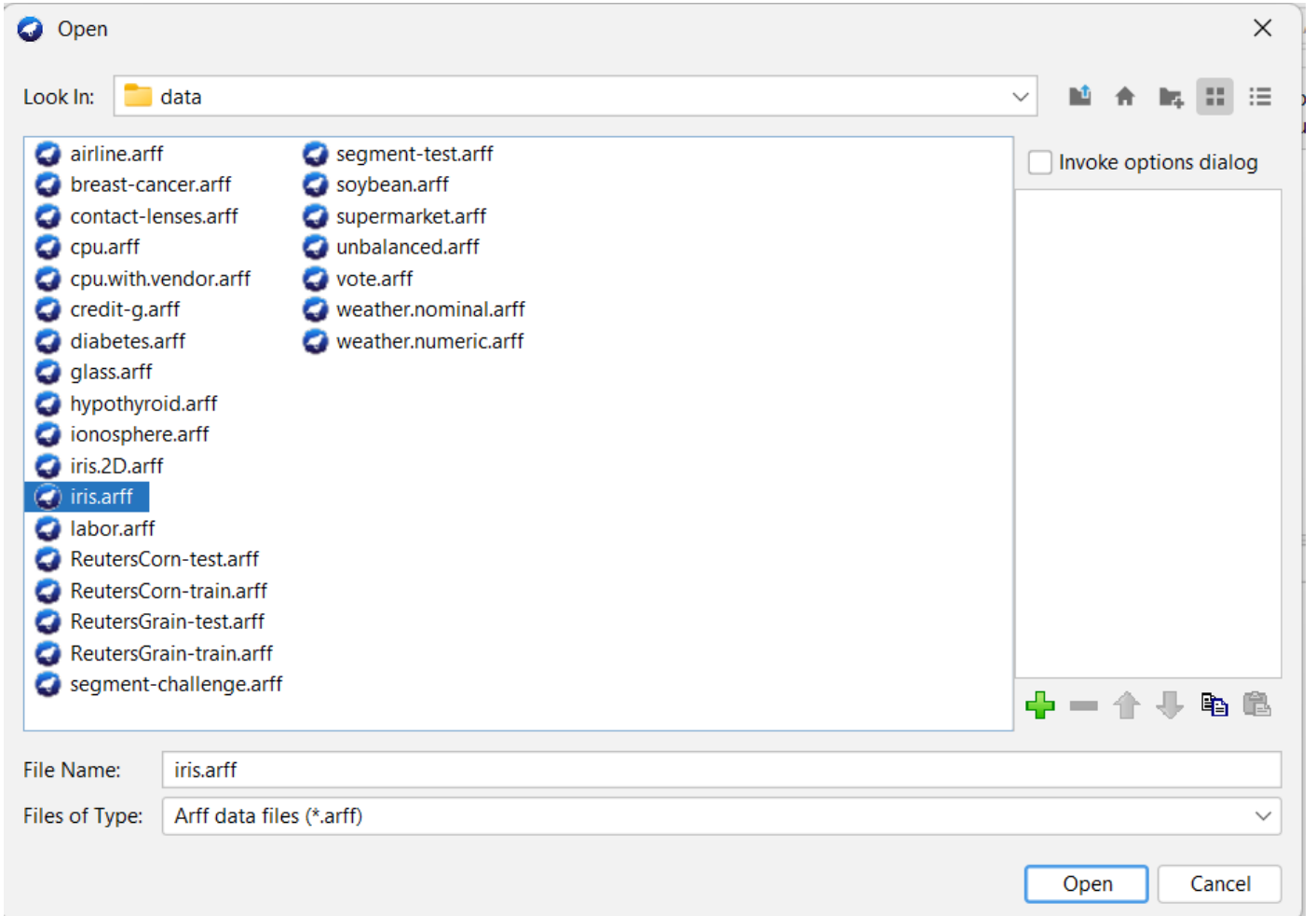
Naive Bayes Algoritması

Naive Bayes sınıflandırıcısının temeli Bayes teoremine dayanır. Lazy (tembel) bir öğrenme algoritmasıdır. Algoritmanın çalışma şekli bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Az bir eğitim verisiyle çok başarılı işler çıkartabilir. Test kümesindeki bir değerın eğitim kümesinde gözlemlenemeyen bir değeri varsa olasılık değeri olarak 0 verir yani tahmin yapamaz. Bu durum genellikle Zero Frequency (Sıfır Frekans) adıyla bilinir. Bu durumu çözmek için düzeltme teknikleri kullanılabilir. En basit düzeltme tekniklerinden biri Laplace tahmini olarak bilinir. Kullanım alanlarına örnek olarak gerçek zamanlı tahmin, çok sınıflı tahmin, metin sınıflandırması, spam filtreleme, duyarlılık analizi ve öneri sistemleri verilebilir.

Weka ile Sınıflandırma Sonuç ve Önerileri

1)- KNN Algoritması ile veri sınıflandırma

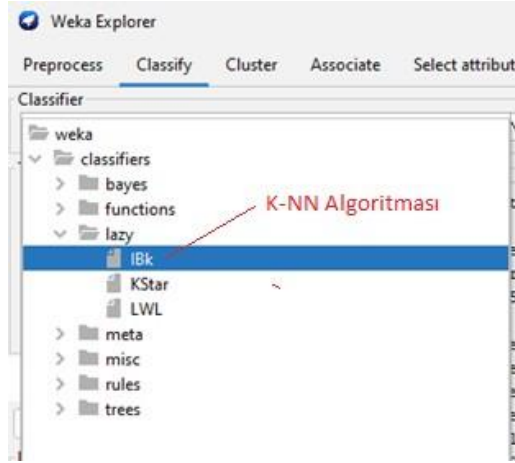
- Öncelikle Preprocess- Open File kısmından Weka'nın içerisinde weka-data dosyası içinden iris veri setine ait iris.arff dosyamızı açıyoruz.(dosya .arff formatında olduğu için dönüştürme işlemi yapmamıza gerek bulunmamaktadır.)



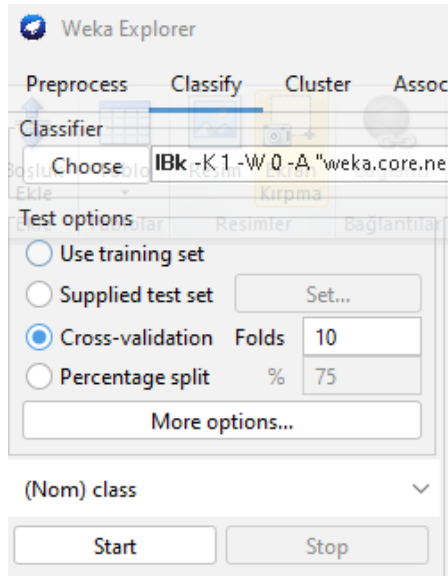
- Daha sonra sınıflandırma işlemi için CLASSIFY bölümüne geçip CHOOSE düğmesini tıkladıktan sonra weka\classifiers\lazy\IBk (K-NN algoritmasını seçiyoruz.)

Şema: weka.classifiers.lazy.IBk

IBk (K-NN algoritması) %96.57 duyarlılık oranına ayrıca doğruluk oranı (%86.14) en fazla olan algoritmalarından biri olmasından dolayı K-NN algoritmasını tercih etmiş oldum.



Algoritmayı çalıştırmadan önce test options kısmına biraz göz atarak ilerleyim



Use Training Set: Kullanılan veri setindeki örneklerinin ne kadar iyi sınıflandırıldığını kontrol eder.

Supplied Test Set : Set kısmından bizim yükleyeceğimiz bir dosya için ne kadar iyi sınıflandırılma yapıldığını kontrol eder.

Cross-validation : Folds alanında girilen değere göre Örneğin 10 girilmişse veri setinin %10 ile test, %90 ile de eğitim yapar.

Percentage split: Vereceğimiz yüzdeye göre sınıflandırmanın ne kadar iyi olduğunu test eder.

Ben Cross-validation seçeneğini işaretleyip Folds değerine 10 vererek devam edeceğim. Weka veri setimizin %10 luk kısmını test %90 luk kısmını da eğitim için kullanırım. Start diyerek algoritmamızı çalıştıralım.

The screenshot shows the Weka Explorer window with the 'Classify' tab selected. The classifier chosen is 'IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\'weka.core.EuclideanDistance -R first-last\'"'. The test options are set to 'Cross-validation' with 'Folds' set to 10. The result list shows two entries: '22:12:08 - lazy.IBk' and '22:12:47 - lazy.IBk'. The classifier output is displayed on the right, showing the run information and the stratified cross-validation summary.

Classifier output

```

=== Run information ===

Scheme:      weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\'weka.core.EuclideanDistance -R first-last\'"
Relation:    iris
Instances:   150
Attributes:  5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      143           95.3333 %
Kappa statistic                    0.93
Mean absolute error                 0.0399
Root mean squared error            0.1747
Relative absolute error             8.9763 %
Root relative squared error        37.0695 %
Total Number of Instances          150
  
```

Yukarıda görüldüğü gibi KNN algoritmamız Normal datayı başarılı bir şekilde sınıflandırma işlemini gerçekleştirdi.

-Classifier Output(Sınıflandırma işlemi sonucunda oluşan çıktılarımız) kısmını aşağıdaki gibi inceleyebiliriz.

The detailed view of the classifier output is shown below, with two sections highlighted by green boxes and numbered 1 and 2.

Section 1: Run information

```

=== Run information ===

Scheme:      weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\'weka.core.EuclideanDistance -R first-last\'"
Relation:    iris
Instances:   150
Attributes:  5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:   10-fold cross-validation
  
```

Section 2: Classifier model (full training set)

```

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      143           95.3333 %
Kappa statistic                    0.93
Mean absolute error                 0.0399
Root mean squared error            0.1747
Relative absolute error             8.9763 %
Root relative squared error        37.0695 %
Total Number of Instances          150
  
```

1)- **Run Information** kısmında veri seti için hangi algoritmayı kullandığımız (IBK->K-NN), veri setimizin isminin ne olduğu(iris), kaç adet veri içerdiği(150), hangi özellikleri içerdiği(sepal.length,sepal.width,petal.length-petal.width) ve hangi test modelinin (10- fold cross-validation) seçildiğine dair bilgiler yer almaktadır.

2)- **Classifier Model** kısmında sınıflandırma işleminin ne şekilde olacağı, nereden başlayacağı hangi aralıklara göre gerçekleşeceği gibi bilgiler yer almaktadır

3)- **Stratified cross-validation** numaralı kısımda seçtiğimiz test modelinde K-NN tahminsel performans sonuçlarını görebiliriz. Test modeli olarak Cross-validation seçtiğimiz için bu başlığa göre sonuçlandırılmıştır.


K-NN normal datanın sınıflandırılmasını yaptım. %75 data oranı ile yapılmıştır. %25 test datası kaldı.

Normal veri seti sonuçlara göre 150 adetlik veri setimizin 143 tanesi doğru, 7 tanesinin yanlış şekilde sonuçlandığını ve %95.3333 başarılı elde edildiğini görebiliriz. Mutlak hata ortalaması: 0.0399 , karesel hata ortalaması: 0.1747, Göreceli mutlak hata: 8.9763

```
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      143           95.3333 %
Kappa statistic                    0.93
Mean absolute error                 0.0399
Root mean squared error             0.1747
Relative absolute error             8.9763 %
Root relative squared error         37.0695 %
Total Number of Instances          150

Log  x 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
      0,940    0,040    0,922    0,940    0,931    0,896    0,952    0,887    Iris-versicolor
      0,920    0,030    0,939    0,920    0,929    0,895    0,947    0,894    Iris-virginica
Weighted Avg.   0,953    0,023    0,953    0,953    0,953    0,930    0,966    0,927

=== Confusion Matrix ===

 a  b  c  <-- classified as
50  0  0 | a = Iris-setosa
 0 47  3 | b = Iris-versicolor
 0  4 46 | c = Iris-virginica
```

Iris-Setosa : **Mavi**

Iris-versicolor: **Kırmızı**

Iris-virginica: **Yeşil**

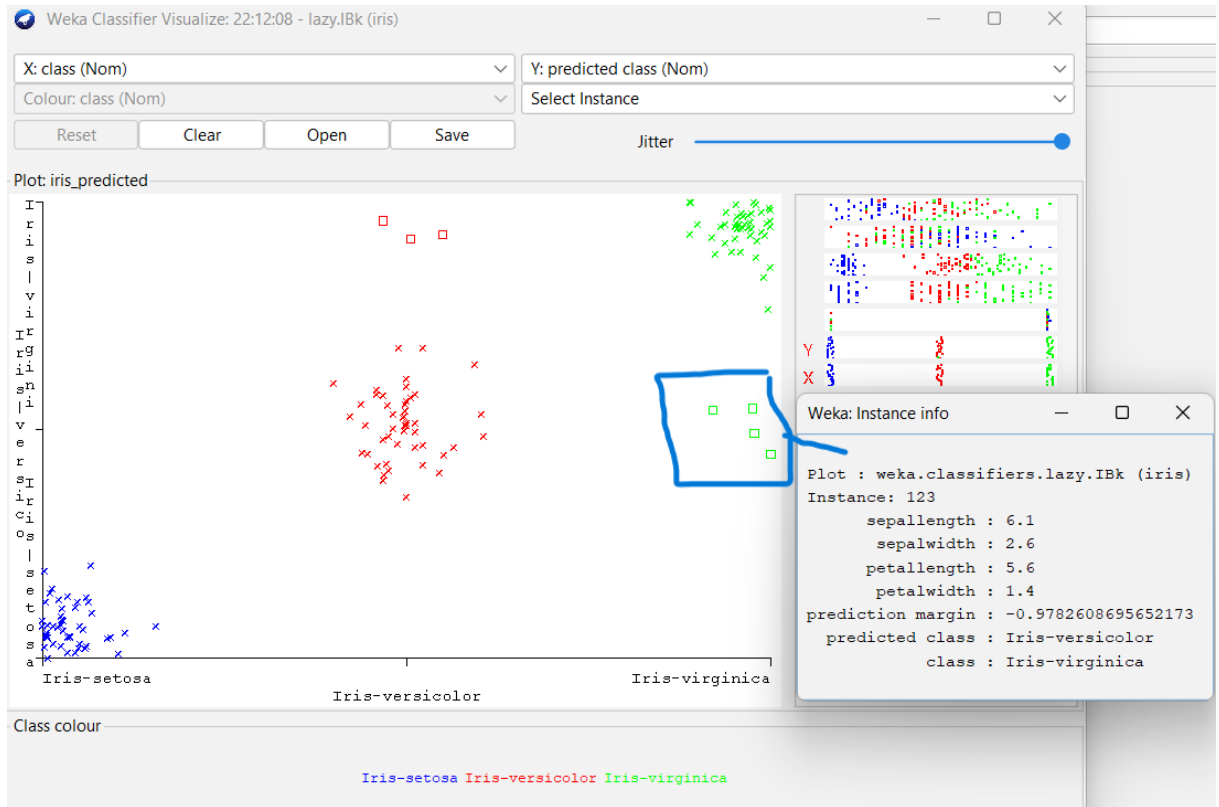
- Aynı zamanda weka üzerinden seçtiğimiz K-NN normal veri setindenki algoritmaya göre oluşansınıflandırma hatalarını da görebilmemiz mümkün .

Bunun için de aynı şekilde **Result** list kısmında algoritmamıza sağ tıkladıktan sonra bu sefer **visualize classifier errors** seçeneğini seçiyoruz.

Yukarıdaki grafikte ise x ile gösterilenler doğru sınıflandırılmış verileri kare şeklinde gösterilen yerler ise yanlış sınıflandırılmış verileri ifade ediyor.

Yukarıdaki tabloda Iris-virginica kısmında bulunan **yeşil bir kare** bu değer in Iris-virginica sınıfında olduğunu ama yanlış şekilde sınıflandırılarak başka bir sınıfa dahil edildiğini gösteriyor.

Karelerden bir tanesinin üzerine çift tıklayarak ayrıntılı bir şekilde görebilirsiniz. Ben Iris-virginica kısmında bulunan yeşil karelerden birine çift tıkladım.



Bu şekilde seçtiğimiz değer için tercih edilen sınıfın (prediction class) Iris-versicolor olduğunu ancak aslında bu değer in(class) Iris-virginica sınıfına ait olduğunu görebiliriz.

Aynı zamanda değere ait özellik (üst yaprak uzunluğu-genişliği, alt yaprak uzunluğu-genişliği)bilgilerine de ulaşabiliriz.

K-NN Test datanın sınıflandırılmasını yaptım. %25 data oranı ile yapılmıştır. %75 testnormal data kaldı.

Test veri seti sonuçlara göre 150 adetlik veri setimizin 143 tanesi doğru, 7 tanesinin yanlış şekilde sonuçlandığını ve %95.3333 başarılı elde edildiğini görebiliriz. Mutlak hata ortalaması: 0.0399 , karesel hata ortalaması: 0.1747, Göreceli mutlak hata: 8.9763

-Classifier Output test data (Sınıflandırma işlemi sonucunda oluşan çıktılarımız) kısmını aşağıdaki gibi inceleyebiliriz

The screenshot shows the Weka Explorer interface. The 'Classify' tab is selected. The 'Classifier' dropdown is set to 'IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"''. The 'Test options' section has 'Cross-validation' selected with 'Folds' set to 10 and 'Percentage split' set to 75. The 'Classifier output' pane shows the following information:

```
=== Run information ===  
  
Scheme:      weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"  
Relation:     iris  
Instances:    150  
Attributes:   5  
              sepalwidth  
              sepalwidth  
              petalwidth  
              petalwidth  
              class  
Test mode:    10-fold cross-validation  
  
=== Classifier model (full training set) ===  
  
IB1 instance-based classifier
```

The screenshot shows the 'Summary' and 'Detailed Accuracy By Class' sections of the Weka Explorer interface. The 'Summary' section shows the following results:

Metric	Value	Percentage
Correctly Classified Instances	143	95.3333 %
Kappa statistic	0.93	
Mean absolute error	0.0399	
Root mean squared error	0.1747	
Relative absolute error	8.9763 %	
Root relative squared error	37.0695 %	
Total Number of Instances	150	

The 'Detailed Accuracy By Class' section shows the following results:

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
0,940	0,040	0,922	0,940	0,931	0,896	0,952	0,887	Iris-versicolor
0,920	0,030	0,939	0,920	0,929	0,895	0,947	0,894	Iris-virginica
Weighted Avg.	0,953	0,023	0,953	0,953	0,953	0,930	0,966	

The 'Confusion Matrix' section shows the following results:

```
=== Confusion Matrix ===  
  
a b c <-- classified as  
50 0 0 | a = Iris-setosa  
0 47 3 | b = Iris-versicolor  
0 4 46 | c = Iris-virginica
```

Iris-Setosa : Mavi

Iris-versicolor: Kırmızı

Iris-virginica: Yeşil

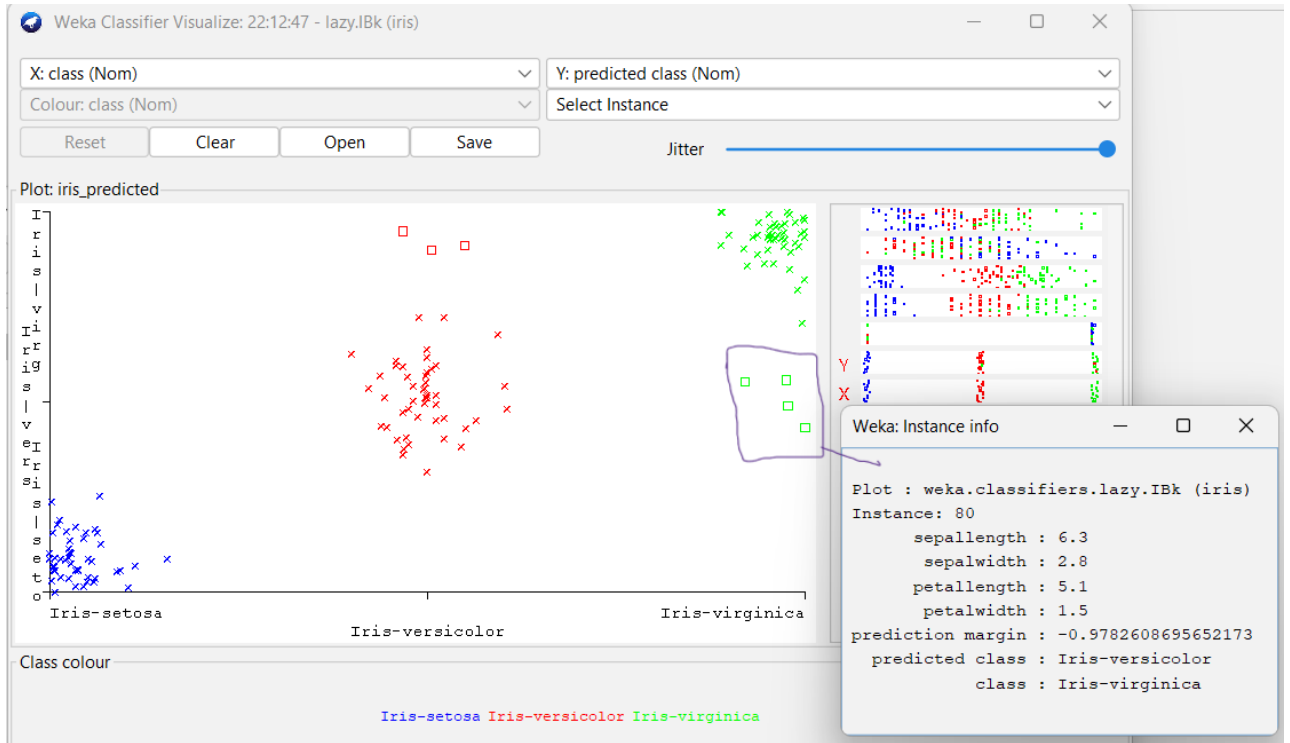
Aynı zamanda weka üzerinden seçtiğimiz K-NN Test veri setindenki algoritmaya göre oluşan sınıflandırma hatalarını da görebilmemiz mümkün .

Bunun için de aynı şekilde **Result** list kısmında algoritmamıza sağ tıkladıktan sonra bu sefer **visualize classifier errors** seçeneğini seçiyoruz.

Yukarıdaki grafikte ise x ile gösterilenler doğru sınıflandırılmış verileri kare şeklinde gösterilenler ise yanlış sınıflandırılmış verileri ifade ediyor.

Yukarıdaki tabloda Iris-virginica kısmında bulunan yeşil bir kare bu değerin Iris-virginica sınıfında olduğunu ama yanlış şekilde sınıflandırılarak başka bir sınıfa dahil edildiğini gösteriyor.

Karelerden bir tanesinin üzerine çift tıklayarak ayrıntılı bir şekilde görebilirsiniz. Ben Iris-virginica kısmında bulunan yeşil karelerden birine çift tıkladım.

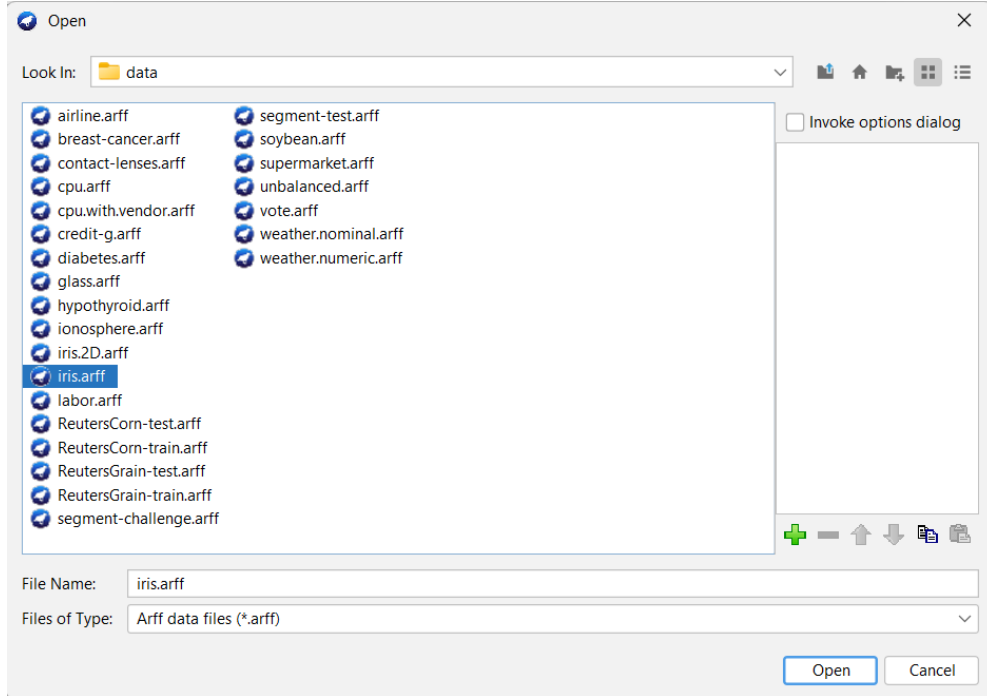


Bu şekilde seçtiğimiz değer için tercih edilen sınıfın (prediction class) Iris-versicolor olduğunu ancak aslında bu değer (class) Iris-virginica sınıfına ait olduğunu görebiliriz.

Aynı zamanda değere ait özellik (üst yaprak uzunluğu-genişliği, alt yaprak uzunluğu-genişliği) bilgilerine de ulaşabiliriz.

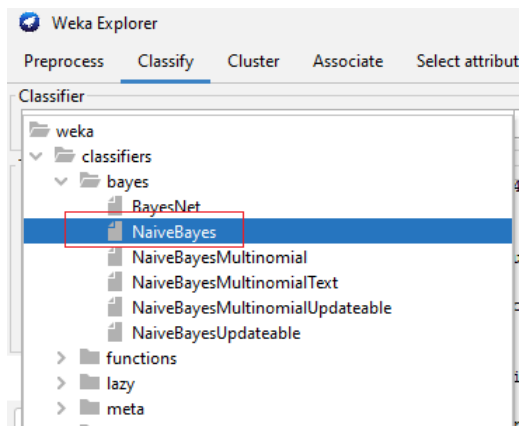
2)- Naive Bayes normal datanın sınıflandırılması

Preprocess- Open File kısmından Weka'nın içerisinde weka-data dosyası içinden iris veri setine ait iris.arff dosyamızı açıyoruz.(dosya .arff formatında olduğu için dönüştürme işlemi yapmamıza gerek bulunmamaktadır.)



- Daha sonra sınıflandırma işlemi için CLASSIFY bölümüne geçip CHOOSE düğmesine tıkladıktan sonra weka > classifiers > bayes > NaiveByes (NaiveByes algoritmasını seçiyoruz.)

Şema: weka.classifiers.bayes.BayesNet



KNN sınıflandırma yöntemi ile Naive Bayes sınıflandırma yönetimini karşılaştırdığımızda normal ve test datasındaki çıkan sonuçlar aşağıdaki gibi elde edilmiştir

Naive Bayes- Normal data

The screenshot shows the Weka Explorer interface with the Naive Bayes classifier selected. The classifier output pane displays the following information:

```

=== Run information ===

Scheme:      weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 1 -S BAYES -E weka.cl
Relation:     iris
Instances:    150
Attributes:   5
  sepalength
  sepalwidth
  petallength
  petalwidth
  class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Bayes Network Classifier
not using ADTree
#attributes=5 #classindex=4
Network structure (nodes followed by parents)
sepalength(3): class
sepalwidth(3): class
petallength(3): class
petalwidth(3): class
class(3):
LogScore Bayes: -481.00632967833803
LogScore BDeu: -525.3834868062277
LogScore MDL: -536.5317339418378
LogScore ENTROPY: -471.39347511858665
LogScore AIC: -497.39347511858665
  
```

The Test options pane shows Cross-validation with 10 folds selected. The Result list pane shows the classifier was run at 22:39:58 on the 'Normal data' set.

```

=== Summary ===

Correctly Classified Instances      139      92.6667 %
Kappa statistic                    0.89
Mean absolute error                 0.0454
Root mean squared error            0.1828
Relative absolute error            10.2111 %
Root relative squared error        38.7793 %
Total Number of Instances          150

=== 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
      0,880    0,050    0,898    0,880    0,889      0,834    0,971    0,906    Iris-versicolor
      0,900    0,060    0,882    0,900    0,891      0,836    0,970    0,919    Iris-virginica
Weighted Avg.   0,927    0,037    0,927    0,927    0,927      0,890    0,980    0,942

=== Confusion Matrix ===

  a  b  c  <-- classified as
50  0  0 | a = Iris-setosa
 0 44  6 | b = Iris-versicolor
 0  5 45 | c = Iris-virginica
  
```

Naive Bayes- Test data

The screenshot shows the Weka Explorer interface with the Naive Bayes classifier selected. The 'Test options' section is set to 'Cross-validation' with 10 folds. The 'Classifier output' section displays the following information:

```
==== Run information ====
Scheme:      weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 1 -S BAYES -E weka.cl
Relation:     iris
Instances:    150
Attributes:   5
  sepallength
  sepalwidth
  petallength
  petalwidth
  class
Test mode:    10-fold cross-validation

==== Classifier model (full training set) ====

Bayes Network Classifier
not using ADTree
#attributes=5 #classindex=4
Network structure (nodes followed by parents)
sepallength(3): class
sepalwidth(3): class
petallength(3): class
petalwidth(3): class
class(3):
LogScore Bayes: -481.00632967833803
LogScore BDeu: -525.3834868062277
LogScore MDL: -536.5317339418378
LogScore ENTROPY: -471.39347511858665
LogScore AIC: -497.39347511858665
```

The 'Result list' on the left shows the test results for the 'Test Data' set, with the entry '22:57:39 - bayes.BayesNet' highlighted.

The 'Summary' section shows the following results:

Metric	Value
Correctly Classified Instances	139
Kappa statistic	0.89
Mean absolute error	0.0454
Root mean squared error	0.1828
Relative absolute error	10.2111 %
Root relative squared error	38.7793 %
Total Number of Instances	150

The 'Detailed Accuracy By Class' section shows the following results:

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
0,880	0,050	0,898	0,880	0,889	0,834	0,971	0,906	Iris-versicolor
0,900	0,060	0,882	0,900	0,891	0,836	0,970	0,919	Iris-virginica
Weighted Avg.	0,927	0,037	0,927	0,927	0,927	0,890	0,980	

The 'Confusion Matrix' section shows the following results:

a	b	c	-- classified as
50	0	0	a = Iris-setosa
0	44	6	b = Iris-versicolor
0	5	45	c = Iris-virginica

Kaynaklar

<https://www.cs.waikato.ac.nz/ml/weka/>

<https://www.softwaretestinghelp.com/weka-datasets/>

<https://docs.huihoo.com/weka/ExperimenterTutorial.pdf>

<https://www.geeksforgeeks.org/building-naive-bayesian-classifier-with-weka/>

<https://arslaney.medium.com/makine-%C3%B6%C4%9Frenmesi-knn-k-nearest-neighbors-algoritmas%C4%B1-bdfb688d7c5f>