KARAR AĞACI SINIFLANDIRMASI

Veri Madenciliği

Samet Kocabaş
150101028

Proje Giriş

Projenin Amacı Nedir ?

Projenin amacı, karar ağacı sınıflandırması yönetim kullanılarak, veri setlerini nitelik ve değerlerine göre sınıflandırarak bir model oluşturmak ve oluşturulan modeli kullanarak girilecek yeni verilerin değerlerine göre bir tahminde bulunmasını sağlamaktır. Sadece belirlediğimiz veri seti için değil, oluşturulan karar ağacı sınıfımıza göre düzenlenebilen her veri seti için sınıflandırma işlemi yapılabilecek bir algoritma geliştirilmiştir. Aslında bir karar ağacı sınıflandırması için basit bir kütüphane oluşturacağız.

Karar Ağacı Sınıflandırması Nedir ?

Karar ağacı öğrenimi istatistik, veri madenciliği ve makine öğreniminde kullanılan öngörülü modelleme yaklaşımlarından biridir. Sınıflama, özellik ve hedefe göre karar düğümleri ve yaprak düğümlerinden oluşan ağaç yapısı formunda bir model oluşturan bir sınıflandırma yöntemidir. Karar ağacı algoritması, veri setini küçük ve hatta daha küçük parçalara bölerek geliştirilir.

Hangi Verisetini Kullanacağız ?

Hastanın aldığı belirli niteliklere göre randevuya gidip gitmediğini gösteren, tıbbi randevu sistem veri setidir. 110.527 tane talep verisi ve 14 tane nitelik bulunur. Biz işimize yaramayacak verileri (gerçekte sonucu etkilemeyecek veriler, hasta id, vb.) çıkartınca, ve hedef niteliğimiz hariç kalan 8 nitelik üzerinden değerlendirme yapacağız. Veri setimiz kaggle.com'dan alınmıştır.

Veriseti Girişleri ve Hedefi Nelerdir. ?

1- Cinsiyet (F/M)
2- Yaş
3- Sigortalı (0/1)
4- Hipertansiyonlu (0/1)
5- Diyabetli (0/1)
6- Alkol kullanıyor (0/1)
7- Özür/Engel (0/1)
8- Mesaj Aldı (0/1)
9- Randevuya Geldi (Yes/No) Sonuç niteliğidir.

Veri seti Hakkında daha fazla detaylar için
https://www.kaggle.com/joniarroba/noshowappointments

Proje Gelişim ve Tasarım

Problemimiz için Kullanılacak Algoritmalar ve Özellikleri Nelerdir?

Karar ağaçlarında, dallanma veya bölümlendirme işlemi için kullanılan algoritmalar 3 gruba ayrılırlar:

1- Entropiye dayalı algoritmalar
2- Sınıflandırma ve regresyon ağaçları
3- Bellek tabanlı algoritmalar

ID3 ve C4.5 Algortimaları Nedir?

Bizim kullanacağımız algoritma, ID3 Algoritması ve C4.5 Algoritmaları,J.R. Quinlan, tarafından 1986 yılında bir veri setinden "karar ağacı" üretmek için geliştirilen, entropiye dayalı algoritmalar.

ID3 algoritması kökten alt dallara doğru ve greedy search (sonuca en yakın durum) teknikleri kullanılır. ID3 algoritması Entropi ve Kazanç Ölçütü üzerine inşa edilmiştir.

C4.5 algoritması ID3 algoritmasının bir uzantısıdır ve sayısal değerlere sahip niteliklerin de karar ağaçlarını oluşturma olanağını sağlamıştır.

Entropi Nedir?

Entropi: rastgeleliğe, belirsizliği ve beklenmeyen durumun ortaya çıkma olasılığını gösterir. Eğer örnekler tamamı düzenli ise entropisi sıfır olur. Eğer değerler birbirine eşit ise entropi 1 olur. Entropi sadece hedef üzerine hesaplanmaz. Ayrıca özellikler üzerine entropi hesaplanabilir. Fakat özellikler üzerine entropi hesaplanırken hedefte göz önüne alır.

Kazanç Ölçütü (Bilgi Kazanımı) Nedir?

Kazanç Ölçütü (Bilgi kazanımı), bir veri setini bir özellik üzerinde böldükten sonra tüm entropiden çıkarmaya dayanır. Entropinin küçük değer içermesi durumunda özelliğin önemi Karar ağacı algoritması ID3 için artmaktadır. Diğer taraftan 1'e yaklaştıkça özelliğinin önemi azalır. Ancak kazanç ölçütünde olay tam tersidir ve bu açıdan entropinin tersi gibi düşünülebilir. Karar ağacı inşa edilirken en yüksek kazanç değerlerine sahip özellik seçilir.

Tasarım,İçerik ve Çözüm

. Projemiz, python kodu ile spyder IDE'si üzerinde yazılacak ve tanımlamalarından bahsettiğimiz entropiye dayalı algortimalar olan ID3 ve C4.5 algoritmalarını kullanılarak basit bir karar ağacı sınıflandırması kütühanesi oluşturacağız.

Bu işlemi gerçekleştirebilmek için DTClassification.py ve DataMining.py adında iki dosyamız olacak. Ilkinde, algoritmatlarımızı kullanarak oluşturacağımız DecisionTreeClassification sınıfı, ikincisinde ise gerekli dosyaları (ilk oluşturduğumuz kütüphane ve veri setimiz) import edip düzenleme ve eğitim yaptığımız alanımız olacak. Öncelikle sınıflandırma yapacağımız DecisionTreeClassification için oluşturmamız gereken fonskiyonlar:

1- createTreeModel(data,labels) :

Bu fonskiyon modelimizin oluşturulduğu özyinelemeli bir fonksiyondur. Modelimizi oluşturmak için ağacın öncelikle kökünü bulup, özyineleme ile dallanma sağlayacak ve bize bir dictonary tipinde bir ağaç yapısı oluşturulacak. Data, niteliklerin ve hedefin (son sıradaki nitelik olarak verilmelidir.) liste olarak verilen,labels ise niteliklerin isimlerinin (hedef hariç) string liste olarak verilen fonskiyonun parametreleridir.

2- calcEntropy(data):

Verilen tek bir niteliğin entropisini hesaplayan fonksiyondur. Data parametresi, bir niteliğin değerlerini liste olarak temsil eder.

3- splitData(data, bestF, val):

Kullanılan ve ya kontrol edilen niteliklerin, tekrar etmemesini sağlamak için o niteliğin veri setinden çıkarılmasını sağlayan fonksiyondur. Data, liste olarak verisetini, bestF, çıkarılması istenen niteliğin indexini ve val değeri ise niteliğin değerini ifade eden parametlerdir.

4- findBestFeature(data):

Bu fonksiyon, bize kazanç ölçütünde en yüksek kazanç değerini sağlayan niteliğin indexini döndürür.data, verisetini temsil eden fonsiyonun parametresidir.

5- majorityCnt(featureList):

Ağaç yapısı oluşturulurken, kökten başlayarak dallandırılırken karşılaştırılan niteliklerin, sadece bir tane kaldığı durumda işlemi sonuç döndüren ve derinlik işlemini sonlandıran fonksiyondur. featureList parametresi, son kalan niteliğin liste olarak ifade eder.

6- predict(model,label, testData):

createTreeModel() ile oluşturduğumuz modelimizi kullanarak, tahmin gerçekleştiren fonksiyondur.model, oluşturulan modeli (dictionary

tipinde),label nitelik isimlerini ve testData ise tahmin edilecek değerlerin liste olarak verileceği parametrelerdir.

1'de bahsedilen fonskiyon ana fonskiyondur.

2-5 arasında bahsedilen fonskiyonlar ana fonksiyonun (createTreeModel) ve ya birbirlerinin çağırması ile çalışan fonksiyonlardır.

6'da bahsedilen fonksiyon, ilk 5'inden bağımsız olarak verilen modele göre tahmin döndüren fonksiyondur.

Bu fonksiyonlar kullanılarak DecisionTreeClassification sınıfımızı oluştuyoruz.

Diğer DataMining.py içeriğimize ise ilk DTClassification.py dosyamızı import ediyoruz. Burada, veri setimizi oluşturduğumuz sınıfta ağaç modelimizi oluşturabilmek için import ediyor ve gereksiz verilerimiz çıkartarak düzenli bir hale getiriyoruz.

Oluşturduğumuz sınıf aslında ID3 algoritması için kusursuz çalışacaktır ama bizim veri setimizde sayısal değerlikli nitelikler olduğu için (yaş niteliği) bunları C4.5 algoritması ile düzenleme işlemeni burada gerçektiriyoruz.

Yaş verilerini diğer verilerden ayırıp, önce verilerimizde bulunan her yaşın sadece bir kere yazılmasını (benzersiz yapmak) sağlayıp, küçükten büyüğe sıralama yapıyoruz. Daha sonrasında kaç gruba ayrılmak isteniyorsa o sayıya bölerek (biz 2 yapacağımız için direk medyan alıyoruz), bulunan sayıların altında,arasında ve üstünde kalan değerleri string'e çevirip veri setimizi sınıfımızı kullanarak model oluşturmak için hazırlıyoruz.

Veri setimizi liste haline getiriyor ve nitelik isimlerini parametre olarak vererek, modelimizi oluşturuyoruz.

Model Oluşturmak için örnek kod:

```
AgacModel = DecisionTreeClassificaiton(versetimiz,nitelik isimleri)
```

Oluşan modelimizi testi için gerçekleştirilen Kod:

```
AgacModel.predict(AgacModel.model,['Gender','Age','Scholarship','Hipertension','Diabetes','Alcoholism','Handcap','SMS_received'],['F','bigger',0,0,0,0,0,0])
```

Oluşan ağacın uzunluğu nedeniyle Örnek Kod ve Sonuç olarak Karar Ağacı sınıflandırması yapılırken her yerde geçen örnekten ; Hava Durumu Veri Setinin değerlerini paylaşacağım:

Verisetimizi import ediyoruz:

```
data = pd.read_csv('weather.csv',delimiter=';')
```

Verisetimizi model oluşturabilmek için label ve data parametrelerine göre düzenliyoruz göre düzenliyoruz:

```
features = data.drop(["Play"],axis=1)
labels = list(features.columns.values)
dataList = data.values.tolist()
```

Modeli Oluşturmak ve Eğitmek için:

```
tree = DTC.DecisionTreeClassificaiton(dataList,labels)
```

Dictonary olarak Ağaç Modelimiz:

```
{
'Outlook': {
            'Overcast': 'Yes',
            'Sunny':    {
                        'Humidity':     {
                                        'High': 'No',
                                        'Normal': 'Yes'
                                        }
                        },
            'Rain': {
                    'Wind':     {
                                'Strong': 'No',
                                'Weak': 'Yes'
                                }
                    }
            }
}
```

Modelimizin tahmini için:

```
label = list(features.columns.values)
tree.predict(tree.model,['Outlook','Temperature','Humidity','Wind'],['S
unny','Hot','Normal','Weak'])
```

Sonuç: *Prediction is Yes*

Proje Sonuç

Projemiz'de bilinen hava durumu sınıflandırması aldığımız sonuç doğrudur.Bu duruma dayanarak asıl sonuç bulmak istediğimiz veri setimiz içinde sorunsuz çalışıyor ve sağlıklı sonuç elde ediyor diyebiliriz.
Proje istenilen gibi sadece tek veri seti için değil, çoğu veri setini düzenleyerek, sonuç bulabilmemize yardımcı sağlar halde çalışmaktadır.