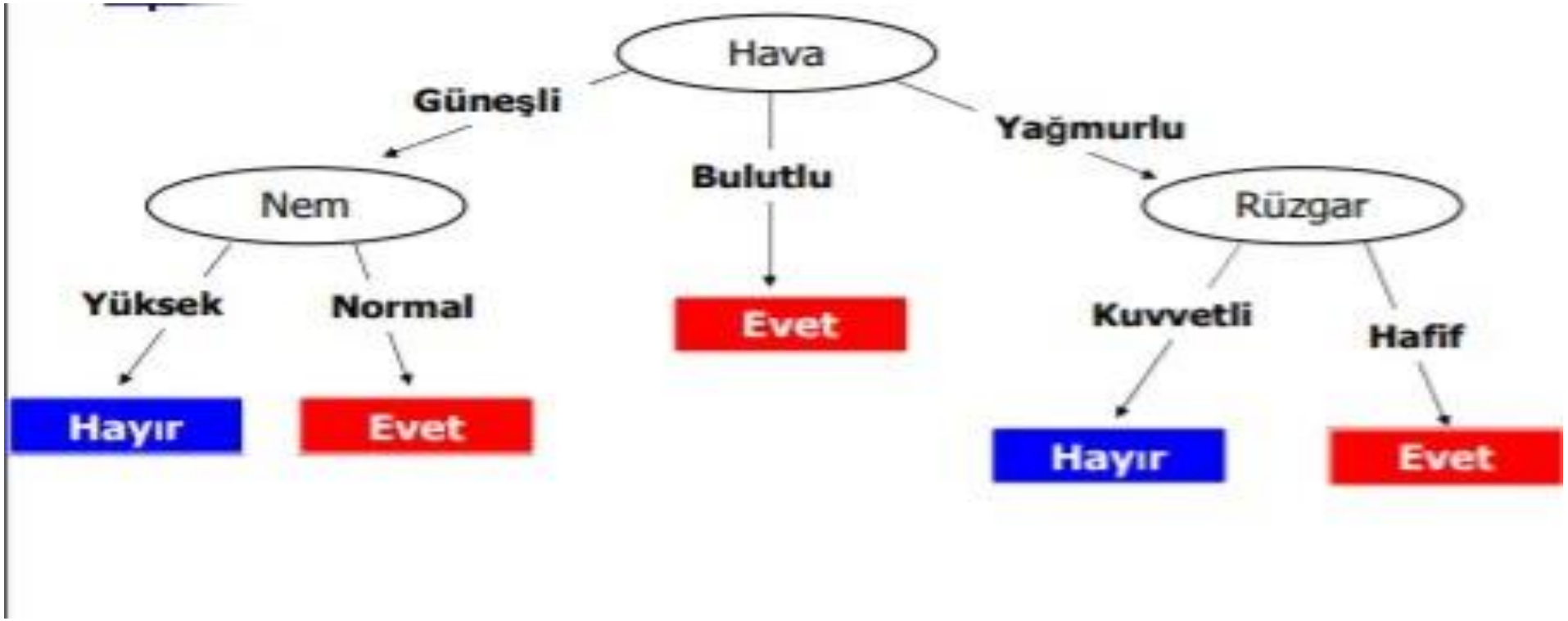
Karar Ağacı (Decision Tree) Algoritması

Karar Ağacı ,

Karar Ağacı ,

temel fikir, giriş verisinin bir kümeleme algoritması yardımıyla tekrar tekrar gruplara bölünmesine dayanır. Grubun tüm elemanları aynı sınıf etiketine sahip olana kadar kümeleme işlemi derinlemesine devam eder.



Karar Ağacı ,

temel fikir, giriş verisinin bir kümeleme algoritması yardımıyla tekrar tekrar gruplara bölünmesine dayanır. Grubun tüm elemanları aynı sınıf etiketine sahip olana kadar kümeleme işlemi derinlemesine devam eder.

Özellikler				Hedef
Hava Durumu	**Sıcaklık**	**Nem**	**Rüzgar**	**Futbol Oyna**
Yağmurlu	Sıcak	Yüksek	Yok	Hayır
Yağmurlu	Sıcak	Yüksek	Var	Hayır
Bulutlu	Sıcak	Yüksek	Yok	Evet
Güneşli	Ilık	Yüksek	Yok	Evet
Güneşli	Soğuk	Normal	Yok	Evet
Güneşli	Soğuk	Normal	Var	Hayır
Bulutlu	Soğuk	Normal	Var	Evet
Yağmurlu	Ilık	Yüksek	Yok	Hayır
Yağmurlu	Soğuk	Normal	Yok	Evet
Güneşli	Ilık	Normal	Yok	Evet
Yağmurlu	Ilık	Normal	Yok	Evet
Bulutlu	Ilık	Yüksek	Var	Evet
Bulutlu	Sıcak	Normal	Yok	Evet
Güneşli	Ilık	Yüksek	Var	Hayır

•Karar Ağacı Algotirmasının adımlarını inceleyecek olursak

- 1.T öğrenme kümesini oluştur 2.T kümesindeki örnekleri en iyi ayıran niteliği belirle
- 3.Seçilen nitelik ile ağacın bir düğümünü oluştur ve bu düğümden
- çocuk düğümleri veya ağacın yapraklarını oluştur. Çocuk
- düğümlere ait alt veri kümesinin örneklerini belirle
- 4. 3. adımda yaratılan her alt veri kümesi için
- ◦ Örneklerin hepsi aynı sınıfa aitse
- ◦ Örnekleri bölecek nitelik kalmamışsa
- ◦ Kalan niteliklerin değerini taşıyan örnek yoksa
- işlem sonlandırılır. Diğer durumda alt veri kümesini ayırmak için 2.
- adımdan devam edilir.
- Algoritması da kendi içinde 2 kategoriye ayrılır. Bu 2 kategoride de birçok algoritma bulunur

1.Entropiye Dayalı Sınıflandırma Ağaçları(ID3, C4.5)
Bu sınıflandırma çeşidini anlamak için yine kavram üzerinden gidelim.

Entropy: rastgeleliğe, belirsizliği ve beklenmeyen durumun ortaya çıkma olasılığını gösterir. Eğer örnekler tamamı düzenli / homojen ise entropisi sıfır olur. Eğer değerler birbirine eşit ise entropi 1 olur.

Entropi formülasyonu:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropi sadece hedef üzerine hesaplanmaz. Ayrıca özellikler üzerine entropi hesaplanabilir. Fakat özellikler üzerine entropi hesaplanırken hedefte göz önüne alır.

Bu durumda entropi formülü:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

Entropi sadece hedef üzerine hesaplanmaz. Ayrıca özellikler üzerine entropi hesaplanabilir. Fakat özellikler üzerine entropi hesaplanırken hedefte göz önüne alır.

Bu durumda entropi formülü:

$$\arg\min_{xj \le x_j^R,\ j=1,\ldots,M} \left[P_l Var(Y_l) + P_r Var(Y_r)\right]$$

Burada Pl ve Pr sırası ile sol ve sağ düğümlerin olasılıklarıdır. M eğitim setindeki değişkenlerin sayılarıdır. Değişken j “xj” olarak gösterilmektedir. R j x ise değişken xj nin en iyi ayırım değerini göstermektedir. Var(Yl), Var(Yr) karşılıklı sağ ve sol alt düğümler için sorumlu vektörlerdir.

$$x_j \leq x_j^R, j = 1, \ldots, M$$

optimal ayırım sorgulaması anlamına gelmektedir. Artıkların karelerini azaltma algoritması Gini ayırım kurallarına benzemektedir. Eğer sınıf k'nın nesneleri değer "1" , diğer sınıfların nesneleri değer "0" ataması yapılır ise, o zaman bu değerlerin örnek varyansı p(k|t)[1- p(k|t)]'e eşit olur. Katışıklık ölçümü i(t) aşağıdaki yolla bulunur

$$i(t) = 1 - \sum_{k=1}^{K} p^2(k \mid t)$$

Burada p(k|t) düğüm t içinde sınıf k'nın koşullara bağlı özelliklerini, K sınıf sayısını, k sınıf indeksini ve t düğüm indeksini göstermektedir

- Örnek Uygulamalar
- #!/usr/bin/env python3
- # -- coding: utf-8 --
- """
- Created on Mon May 3 23:58:38 2021
- @author: deniz
- """
- import numpy as np
- import matplotlib.pyplot as plt
- import pandas as pd
- from sklearn.model_selection import train_test_split
- from sklearn.preprocessing import StandardScaler
- from sklearn.tree import DecisionTreeClassifier
- from sklearn import metrics
- from sklearn.metrics import confusion_matrix
- from matplotlib.colors import ListedColormap

- #veri setini alıyoruz
- data = pd.read_csv("/home/deniz/Masaüstü/decision tree/decision_tree.csv")
- #başlıkları bastırıyoruz
- print(data.head())

- #ilk 4 veri setini ve verinin parametrelerini aldık
- X = data.iloc[:,[2,3]].values
- y = data.iloc[:,4].values

- #eğitim ve test veri setlerini oluşturduk
- X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25, random_state= 0)
- sc_X = StandardScaler()
- X_train = sc_X.fit_transform(X_train)
- X_test = sc_X.transform(X_test)

- #kara ağacı fonksiyonumuzu çağıralım
- classifier = DecisionTreeClassifier()
- classifier = classifier.fit(X_train,y_train)

- #doğruluk payımızın ne kadar olduğunu kontrol edelim
- y_pred = classifier.predict(X_test)
- print('Doğruluk payı', metrics.accuracy_score(y_test,y_pred))
- cm = confusion_matrix(y_test, y_pred)
- print(cm)

- #Veri setlerimiziden oluşturulan modeli görselleştirelim
- X_set, y_set = X_test, y_test
- X1, X2 = np.meshgrid(np.arange(start = X_set[:,0].min()-1, stop= X_set[:,0].max()+1, step = 0.01),np.arange(start = X_set[:,1].min()-1, stop= X_set[:,1].max()+1, step = 0.01))
- plt.contourf(X1,X2, classifier.predict(np.array([X1.ravel(),

- X2.ravel()]).T).reshape(X1.shape), alpha=0.75, cmap = ListedColormap(("white","black")))
- plt.xlim(X1.min(), X1.max())
- plt.ylim(X2.min(), X2.max())
- for i,j in enumerate(np.unique(y_set)):
- plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1], c = ListedColormap(("gray","red"))(i),label = j)
- plt.title("Karar Ağacı")
- plt.xlabel("Tahmini Yaş")
- plt.ylabel("Maaş")
- plt.legend()
- plt.show()

Çıktı :