



INTRODUCTION TO ARTIFICIAL INTELLIGENCE DERSİ

DECISION TREE ÖDEVİ

FATİH TALHA TÜMER – 191180081

İÇİNDEKİLER

İÇİNDEKİLER.....	i
ŞEKİLLER LİSTESİ.....	ii
1. GİRİŞ	1
2 SINIFLANDIRICI	1
3 SONUÇ VE KAZANIMLAR.....	1
KAYNAKÇA	Error! Bookmark not defined.

ŞEKİLLER LİSTESİ

Şekil 1: Rastgele verilerin oluşturulması.....	1
Şekil 2: Kullanıcı tarafından ek verilerin girilmesi	2
Şekil 3: Node sınıfı.....	3
Şekil 4: Veri ön hazırlığı ve modelin eğitilmesi	3
Şekil 5: Eğitilen modelin Graphviz kütüphanesi aracılığıyla ağaç şeklinde görselleştirilmesi .	4

1. GİRİŞ

Bu çalışmada bir karar ağacı algoritması, hazır kütüphaneler kullanılmadan geliştirilmiş, veri rastgele olarak kullanıcının istediği kadar yeni özellik girme özelliği eklenmiş bir algoritma geliştirilmiştir. Information gin için gini index ve entropi değerleri kullanılmıştır. Kodlar, açıklamalar ve görselleştirilmiş ağaç çalışmanın diğer bölümlerinde paylaşılmıştır.

2 SINIFLANDIRICI

- Ödev, Python, jupyter notebook ve Google colab kullanılarak oluşturulmuştur.
- Karar ağacının gerçekleştirilmesinde herhangi bir hazır fonksiyon kullanılmamıştır.

Ödevin oluşturulma aşamaları:

1. İstenilen tablonun, kullanıcının gireceği satır sayısına göre rastgele oluşturulması:

```
# rastgele veri üretimi
row_count = 100
df = generate_df(row_count)
df
```

	yas	cinsiyet	saglik	is	gelir	maas	borc	miras	ev	araba	kredi
0	21	erkek	iyi	isci	orta	15427.0	var	yok	yok	var	hayir
1	72	erkek	hasta	issiz	orta	2061.0	var	var	yok	var	hayir
2	76	kadin	hasta	yonetici	kotu	8815.0	yok	yok	yok	var	hayir
3	28	erkek	hasta	issiz	iyi	12990.0	var	yok	yok	yok	hayir
4	58	erkek	iyi	issiz	kotu	7009.0	yok	yok	var	var	hayir
...
95	39	kadin	hasta	issiz	orta	17718.0	var	var	yok	var	hayir
96	66	erkek	hasta	issiz	orta	6355.0	var	yok	var	yok	hayir
97	41	erkek	iyi	isci	iyi	3418.0	var	yok	var	var	evet
98	22	erkek	iyi	issiz	kotu	13127.0	yok	yok	yok	var	evet
99	34	erkek	hasta	issiz	iyi	447.0	yok	yok	var	var	evet

100 rows x 11 columns

Şekil 1: Rastgele verilerin oluşturulması

Kullanıcının istediği kadar özelliği tabloya eklemesini sağlayan kısım. Kullanıcı önce başlığı sonra da o başlık için girmek istediği kadar örnek girebiliyor. Daha sonra girilen bu değerler dataframe'e ekleniyor.

```
#eklenmek istenen veriler

baslik = input("Eklenmek istenen verinin başlığı:")
i=0
secenekler = []
while True:
    temp = input(baslik + " için " + str(i+1) + ".secenek:")
    if temp == "0":
        break
    else:
        secenekler.append(temp)
    i+=1
ek_veri=[]
for i in range(row_count):
    ek_veri.append(np.random.choice(secenekler))

ek_veri = pd.Series(ek_veri)
df[baslik] = ek_veri
```

```
Eklenmek istenen verinin başlığı:baslik1
baslik1 için 1.secenek:baslik2
baslik1 için 2.secenek:aslik3
baslik1 için 3.secenek:aslik4
baslik1 için 4.secenek:baslik 4
baslik1 için 5.secenek:baslik 5
baslik1 için 6.secenek:0
```

df

	yas	cinsiyet	saglik	is	gelir	maas	borc	miras	ev	araba	kredi	baslik1
0	26	erkek	hasta	isci	kotu	10710.0	var	var	yok	var	evet	aslik4
1	77	erkek	hasta	memur	kotu	11695.0	yok	var	yok	yok	evet	baslik 4
2	30	kadın	hasta	memur	orta	5565.0	var	yok	var	var	hayir	aslik4
3	61	kadın	hasta	issiz	orta	6742.0	yok	var	var	var	hayir	aslik4
4	36	erkek	hasta	issiz	kotu	16957.0	yok	var	yok	yok	hayir	baslik 4

Şekil 2: Kullanıcı tarafından ek verilerin girilmesi

Node sınıfı oluşturuldu. Eğitilecek olan modelin saklanması için ağaç yapısının temeli olması için node yapısı kullanıldı.

```
[3] class Node():
    def __init__(self, feature_index=None, threshold=None, left=None, right=None, info_gain=None, value=None):
        #karar ağacı için değişkenler
        self.feature_index = feature_index
        self.threshold = threshold
        self.left = left
        self.right = right
        self.info_gain = info_gain

        #yaprak node'ları için sonuç değeri
        self.value = value
```

Şekil 3: Node sınıfı

DecisionTree oluşturuldu.

__init__, select_best_split (en yüksek information gain'e sahip spliti bulma),

split (belirlenen özelliğe göre böle işleminin gerçekleşmesi), fit (verilerin eğitime hazırlanması),

predict (dizi şeklinde alınan verilerin make_prediction yardımıyla tahmin edilmesi)

build_tree (verinin minimum sample count ve maximum depth kurallarına uyması durumunda tekrar tekrar node üretilmesi)

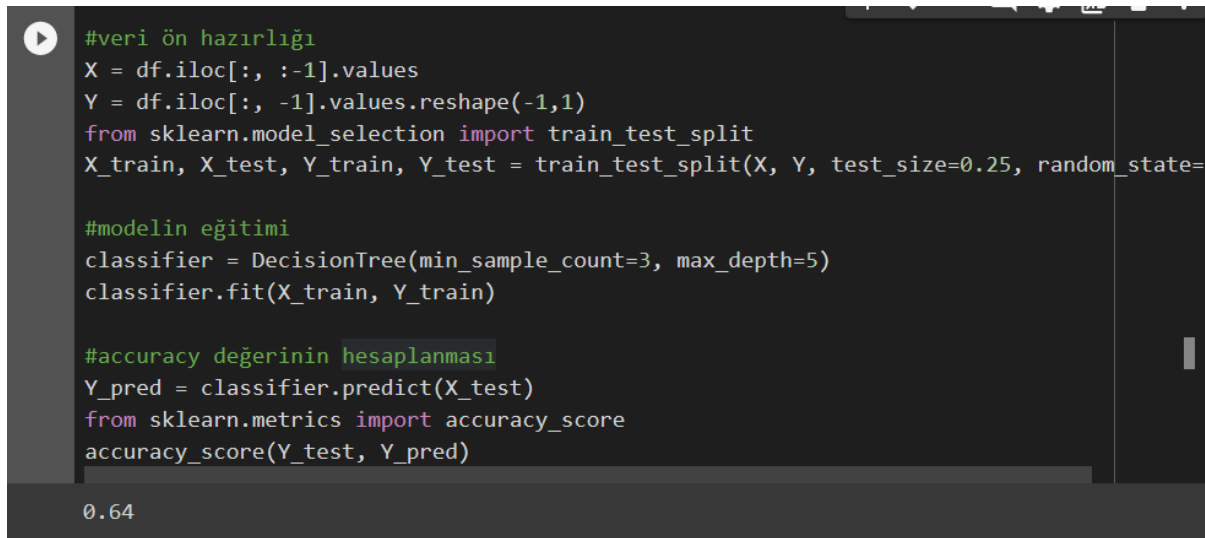
calculate_leaf_node (kararlara ulaşılan son yaprak node'ların ulaştıkları kararların atanması)

feature_type (kategorik ve numerik verilerin ayrı işlemler görmesi için ayırma işlemi)

Entropi (entropinin hesaplanması)

Gini (gini indexinin hesaplanması), info_gain (entropi ya da gini index kullanarak information gain'in hesaplanması) fonksiyonları kullanıldı.

Verinin ön hazırlığı ve modelin eğitilmesi, accuracy değerinin tespit edilmesi



```
#veri ön hazırlığı
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values.reshape(-1,1)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=

#modelin eğitimi
classifier = DecisionTree(min_sample_count=3, max_depth=5)
classifier.fit(X_train, Y_train)

#accuracy değerinin hesaplanması
Y_pred = classifier.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, Y_pred)

0.64
```

Şekil 4: Veri ön hazırlığı ve modelin eğitilmesi

Eğitilen modelin graphviz kütüphanesi kullanılarak görselleştirilmesi (Jupyter notebook'da graphviz kütüphanesinde hata alındığı için çalışma Google colab'de devam ettirilmiştir.) Sonraki sayfada çıktının yüksek çözünürlüklü görüntüsü bulunmaktadır. PDF versiyonu da ödev klasöründe mevcuttur.

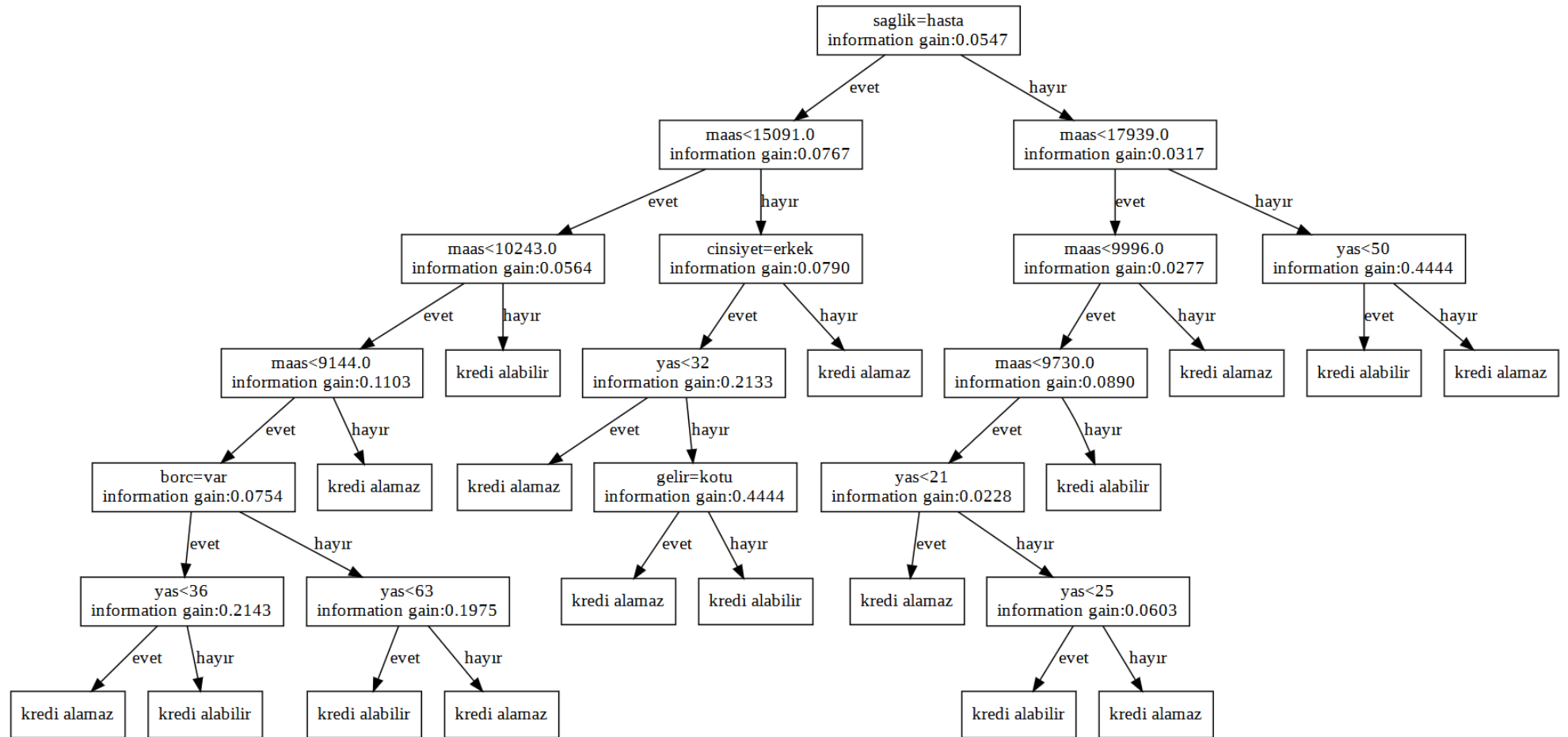
```
[26] from graphviz import Digraph

basliklar = list(df.columns[:-1])
type(basliklar)
def visualize(node, dot=None):
    if dot is None:
        dot = Digraph()
    if str(node.value)=="None":
        if type(node.threshold)==int or type(node.threshold)==float:
            deger = basliklar[node.feature_index] + "<" + str(node.threshold) + "\n" + "information gain:" + "{:.4f}".format(node.info_gain)
        else:
            deger = basliklar[node.feature_index] + "=" + str(node.threshold) + "\n" + "information gain:" + "{:.4f}".format(node.info_gain)

    else:
        if node.value=="evet":
            deger = "kredi alabilir"
        else:
            deger = "kredi alamaz"
    dot.node(str(id(node)), deger, shape='rectangle')
    if node.left:
        dot.edge(str(id(node)), str(id(node.left)), label="evet")
        visualize(node.left, dot)
    if node.right:
        dot.edge(str(id(node)), str(id(node.right)), label="hayır")
        visualize(node.right, dot)
    return dot

# Example usage
node = classifier.root
dot = visualize(node)
dot.render('binary_tree.gv', view=True)
```

Şekil 5: Eğitilen modelin Graphviz kütüphanesi aracılığıyla ağaç şeklinde görselleştirilmesi



3 SONUÇ VE KAZANIMLAR

Karar ağacı yapıları ve ağaç yapılandırılırken kullanılan purity ölçüm değerlerinin (entropi ve gini index) ne kadar önemli ve faydalı olduğu anlaşıldı. 2 yöntem de başarılı bir şekilde eklendi. Hazır kütüphaneler incelendi ve bu kütüphanelerin geliştirme/yayınlama süreçlerinde zorluklar anlaşıldı. Bu kütüphanelerin sağladığı kolaylıkların farkına varıldı. Kod yazma yeteneğinde ve makine öğrenmesini kavramada gelişmeler oldu.