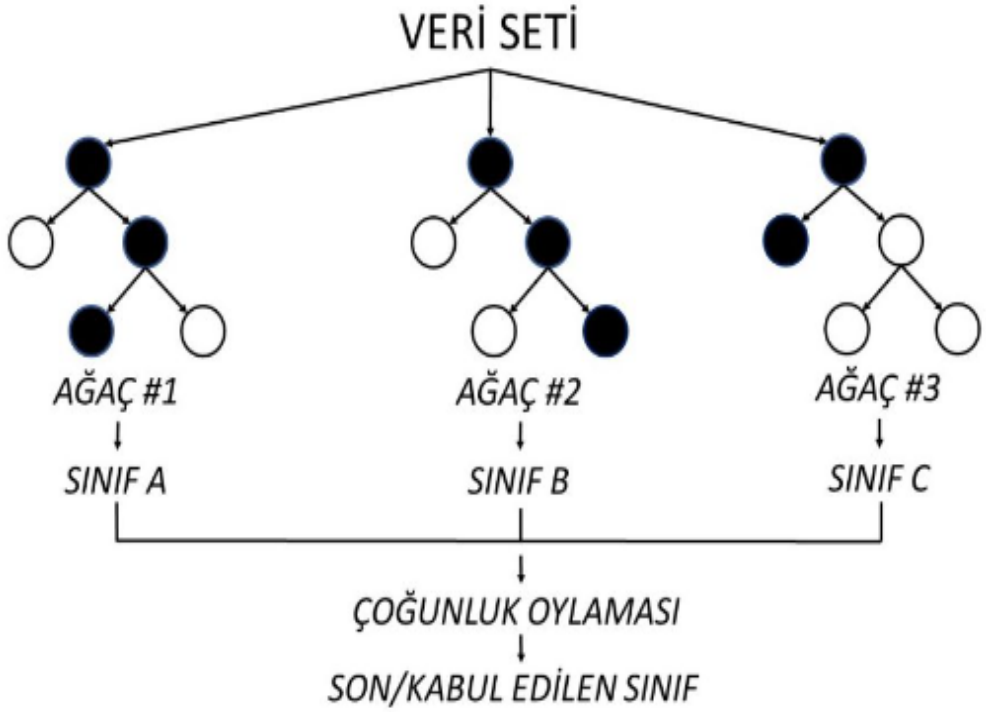
Rastgele Orman (Random Forest) Algoritması Nedir?

Random Forest (***Rastgele Orman***) algoritması; birden çok karar ağacı üzerinden her bir karar ağacını farklı bir gözlem örneği üzerinde eğiterek çeşitli modeller üretip, sınıflandırma oluşturmanızı sağlamaktadır.

Kullanım kolaylığı ve esnekliği; hem sınıflandırma hem de regresyon problemlerini ele aldığı için benimsenmesini ve kullanımının yaygınlaşmasını hızlandırdı.

Algoritmaya yönelik en beğenilen nokta ise; veri kümeniz üzerinde çeşitli modellerin oluşturulması ile kümenizi yeniden ve daha derin keşfetme imkanı sunmasıdır.



Algoritma;

- Analiz edilecek veri seti hazırlanır,
  (*Analiz edilecek küme oluşturulur, gerekli görülürse veri temizlemesi gerçekleştirilir.*)
- Algoritma her bir örnek için karar ağacı oluşturur ve her bir karar ağacının tahmini değer sonucu oluşur,
- Tahmin sonucu oluşan her değer için oylama gerçekleştirilir,
  *(*Sınıflandırma problemi için **Modu** (**Mode**), Regresyon problemi için **Ortalamayı** (**Mean**)*)
- Son olarak algoritma son tahmin için en çok oylanan değeri seçerek sonuç oluşturur.

adımları ile analiz gerçekleştirmektedir.

Rastgele Orman (Random Forest) Uygulama Örneği

RF (Random Forest, Rastgele Orman) algoritması uygulamasına örnek olarak, çokça duyulan ve kullanılan Iris Veri Seti üzerinden basit bir örnek ile çözümlemelerde bulunalım.

İlgili veri setine ait ***csv. uzantılı dosyasına buradan ulaşabilirsiniz.***

Iris Versicolor **Iris Setosa** **Iris Virginica**

Iris Veri Seti 3 Iris bitki türüne (Iris Setosa, Iris Virginica ve Iris Versicolor) ait, her bir türden 50 örnek olmak üzere toplam 150 örnek sayısına sahip bir veri setidir.

Iris Veri Seti içerisinde;

Sınıflar (Türler);

- Iris Setosa,
- Iris Versicolor,
- Iris Virginica.

Veri Özellikleri (Ortak Özellikler);

- Sepal Uzunluk (cm),
- Sepal Genişlik (cm),
- Petal Genişliği (cm)
- Petal Uzunluk (cm).

özellik ve değerleri bulunmaktadır.

Dilerseniz hızlıca analizimizi gerçekleştirmek için adımlarımızı uygulamaya başlayalım.

1-Gerçekleştireceğimiz analizler için kullanacağımız kütüphanelerimizi projemize dahil edelim;
(*Scikit-Learn*, *Pandas*, *Matplotlip*, *Seaborn*)

```
>>> from sklearn import datasets
    import pandas as pan
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn import metrics
    import matplotlib.pyplot as plot
    import seaborn as sea
```

- **datasets**: Kullanacağımız veri kümesini çalışmaya dahil etmek için, (Iris)
- **pan**: İçeri aktardığımız veri kümesinin bir çok boyutlu olarak kullanacağımız veri çerçevesini (DataFrame) oluşturmak için,
- **train_test_split**: Eğitim ve Test kümesi işlemleri kullanımı için,
- **RandomForestClassifier**: Rastgele Orman modelini kullanmak için,
- **metrics**: Tahminlememizde doğruluk hesaplaması kullanmak için,
- **plot**: Görselleştirme kullanmak için,
- **sea**: Görselleştirme kullanmak için

ilgili kütüphaneleri projemize aktarmaktayız.

```
from sklearn import datasets
import pandas as pan
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import matplotlib.pyplot as plot
import seaborn as sea
```

2-Çalışmamıza ilgili verilerimizi dahil edelim ve önizleme gerçekleştirelim;

```
>>> iris_dataset = datasets.load_iris()
    print(iris_dataset)
```

***Önceki çalışmamızda *yerel dizin üzerinden ilgili verilerimizi projemize dahil etmiştik*;

```
Iris_Data = pd.read_csv("D:\Yedekleme\Iris.csv")
```

```
iris_dataset = datasets.load_iris()
```

```
print(iris_dataset)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

3-Etiket türlerimizin ve özelliklerimizin önizlemesini gerçekleştirelim;

```
>>> print(iris_dataset.target_names)
    print(iris_dataset.feature_names)
```

```
print(iris_dataset.target_names)

print(iris_dataset.feature_names)
```

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

4-DataFrame yapımızı oluşturmadan verilerimize yönelik önizleme gerçekleştirelim; (*TOP 10 Kayıt*)

```
>>> print(iris_dataset.data[0:10])
    print(iris_dataset.target)
```

```
print(iris_dataset.data[0:10])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

```
print(iris_dataset.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

5-Iris veri kümesi üzerinden bir DataFrame oluşturalım;

```
>>> salt_data=pan.DataFrame({
    'sepal length':iris_dataset.data[:,0],
    'sepal width':iris_dataset.data[:,1],
    'petal length':iris_dataset.data[:,2],
    'petal width':iris_dataset.data[:,3],
    'species':iris_dataset.target
})
```

```
salt_data=pan.DataFrame({
    'sepal length':iris_dataset.data[:,0],
    'sepal width' :iris_dataset.data[:,1],
    'petal length':iris_dataset.data[:,2],
    'petal width' :iris_dataset.data[:,3],
    'species'     :iris_dataset.target
})
```

6-Oluşturmuş olduğumuz DataFrame'in önzilemesini gerçekleştirelim;

```
>>> salt_data.head()
```

```
salt_data.head()
```

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

7-Veri kümemizi test ve eğitim/öğrenme kümeleri olarak ikiye bölelim-ayıralım; (*Test-Train*)

**Verileri; 35% Test, 65% Eğitim olarak ayıralım.*

***test_size** ile verilerin %kaçının test için kullanılacağını ifade belirleyebilmektesiniz. (Örneğimizde %35.)*

***train_size** ile verilerin %kaçının eğitim için kullanılacağını ifade belirleyebilmektesiniz.*

***shuffle** ile verilerin bölünmeden karıştırma uygulanıp-uygulanmayacağınız belirleyebilmektesiniz.*

***random_state** ile bölünmeden önce verilere-veriye uygulanan karıştırmayı kontrol edebilmektesiniz.*

```
>>> X=salt_data[['sepal length', 'sepal width', 'petal length', 'petal width']]
    y=salt_data['species']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, shuffle=True)
```

```
X=salt_data[['sepal length', 'sepal width', 'petal length', 'petal width']]
y=salt_data['species']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, shuffle=True)
```

8-Veri kümemizi bölme-ayırma işleminden sonra modeli eğitim seti üzerinde eğitip, test seti üzerinde tahminler gerçekleştirelim;

**Verileri; 120 ağaç sayısı olarak ele alalım.*

***n_estimators** ile maksimum oylama veya tahmin ortalamalarını almadan önce inşa etmek istediğiniz ağaç sayısını belirleyebilmektesiniz.*

***min_sample_leaf** ile daha önce bir karar ağacı oluşturduysanız, minimum örnek yaprak boyutunun önemini anlayabilir ve sonrasında değer üzerinden sayısını belirleyebilmektesiniz.*

***min_sample_split** ile dahili düğümü bölmek için gereken minimum örnek sayısını belirleyebilmektesiniz.*

***max_depth** ile ağacın maksimum derinliğini belirleyebilmektesiniz.*

Kullanılmaz ise varsayılan olarak 0, tüm yapraklar saf olana veya tüm yapraklar min_samples_split örneklerinden daha azını içerene kadar düğümler genişletilmektedir.

```
>>> clf=RandomForestClassifier(n_estimators=120)
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_test)
```

```
clf=RandomForestClassifier(n_estimators=120)

clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

9-Eğitimden sonra, gerçek ve tahmin edilen değerleri kullanarak doğruluk değerini kontrol edelim;

```
>>> print("Accuracy Value:",metrics.accuracy_score(y_test, y_pred))
```

```
print("Accuracy Value:",metrics.accuracy_score(y_test, y_pred))
Accuracy Value: 0.9433962264150944
```

10-Eğitim işlemlerimizin tamamlanmasından sonra dilerseniz tahminlemenizi tek bir belirlediğiniz öğe içinde kontrol edebilirsiniz;

- sepal length = 1
- sepal width = 2
- petal length = 3
- petal width = 4

değerleri için;

>>>clf.predict([[1, 2, 3, 4]])
>>> array([1]) değeri elde edilir.

[1]: Versicolor çiçeğini ifade etmektedir.

```
clf.predict([[1, 2, 3, 4]])
```

```
array([1])
```

Şimdi analiz yapımızı bira daha geliştirip, basit seviyede görselleştirme gerçekleştirelim.

11-(8) adım üzerindeki işlemlerimizi tekrar ederek analizimize başlayalım;

>>> clf=RandomForestClassifier(n_estimators=120)
clf.fit(X_train,y_train)

```
clf=RandomForestClassifier(n_estimators=120)
```

```
clf.fit(X_train,y_train)
```

```
RandomForestClassifier(n_estimators=120)
```

12-Rastgele Orman sınıflandırıcımızı bir seviye daha derinleştirip, eğitimde bulunalım;

*__bootstrap__ *ile ağaç oluştururken önyükleme örneklerinin kullanılıp kullanılmadığı kontrol edebilmektesiniz. Alt örnek boyutu; eğer bootstrap=True (varsayılan) ise max_samples parametresi ile kontrol edilmekte, aksi takdirde her bir ağacı oluşturmak için tüm veri seti kullanılmaktadır.*

*__class_weight__ *ile formdaki sınıflarla ilişkili ağırlıkları belirleyebilmektesiniz.*

*__criterion__ *ile bir bölünmenin kalitesini ölçme işlevini belirleyebilmektesiniz. Desteklenen kriterler Gini Safsızlığı/Kirliliği için "gini", bilgi kazancı için "entropi" değerleri kullanılmaktadır.*

*__max_features__ *ile bir düğümü ayırırken dikkate alınacak maksimum özellik sayısını belirleyebilmektesiniz.*

*__max_leaf_nodes__ *ile maksimum kullanılacak yaprak sayısını belirleyebilmektesiniz.*

*__min_impurity_decrease__ *ile safsızlık/kirlilik değerlerini belirleyebilmektesiniz.*

*__min_impurity_split__ *ile safsızlık/kirlilik bölünmesi değerini belirleyebilmektesiniz.*

*__min_weight_fraction_leaf__ *ile bir yaprak düğümde olması gereken ağırlıklar toplamının minimum ağırlıklı değerini belirleyebilmektesiniz.*

*__n_jobs__ *ile paralel olarak çalıştırılacak iş sayısını belirleyebilmektesiniz.*

*__oob_score__ *ile tahmin hatalarını hesaplayabilmektesiniz.*

*__verbose__ *ile ayrıntı düzeyini belirleyebilmektesiniz.*

*__warm_start__ *ile işlem sonucu dahilinde önceki uygun çözümü yeniden kullanmak yerine, yeni bir orman oluşturulmasını sağlayabilmektesiniz. (EK Parametreler ve bilgilendirmeleri.)*

```
>>> RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
    max_depth=None, max_features='auto', max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
    min_samples_leaf=1, min_samples_split=0,
    min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
    oob_score=False, random_state=None, verbose=0,
    warm_start=False)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=0,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
RandomForestClassifier(min_samples_split=0, n_jobs=1)
```

13-Özelliklere yönelik önem puanlarını değerlendirelim;

**sort_values* *ile verilerinizi belirlemiş olduğunuz sütuna/parametreye göre sıralayabilmektesiniz.*

sort_values(by='sıralanacak sütun', ascending=False)

```
>>> feature_imp = pan.Series(clf.feature_importances_,index=iris_dataset.feature_names)
                          .sort_values(ascending=False)
>>> feature_imp
```

```
feature_imp = pan.Series(clf.feature_importances_,index=iris_dataset.feature_names).sort_values(ascending=False)
```

```
feature_imp
```

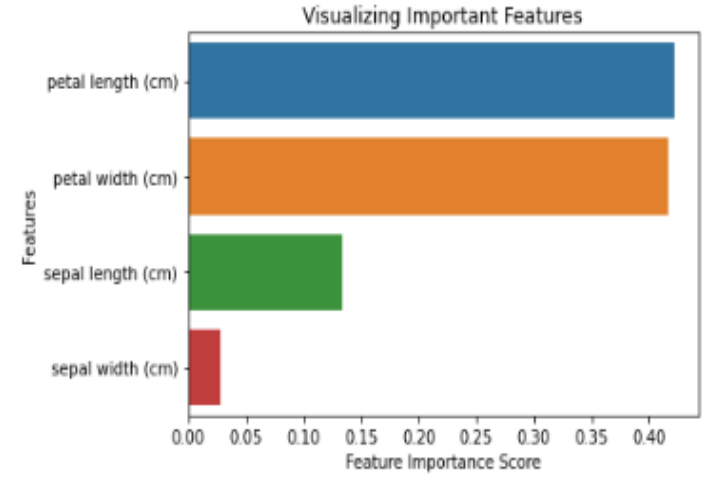
```
petal width (cm)     0.459666
petal length (cm)    0.411183
sepal length (cm)    0.105697
sepal width (cm)     0.023455
dtype: float64
```

14-Çıktı olarak elde ettiğimiz skor değerlerini görselleştirelim;

```
>>> sea.barplot(x=feature_imp, y=feature_imp.index)
    plot.xlabel('Feature Importance Score')
    plot.ylabel('Features')
    plot.title("Visualizing Important Features")
    plot.show()
```

```
sea.barplot(x=feature_imp, y=feature_imp.index)

plot.xlabel('Feature Importance Score')
plot.ylabel('Features')
plot.title("Visualizing Important Features")
plot.show()
```



Skorlar üzerinden çıkarımla düşük öneme sahip olan parametreleri tahminleyicimizde göz ardı edip tekrar analizde bulunabiliriz.

15-Analizimizde kullanacağımız veri kümemizin parametreleri yeniden belirleyelim; (*Test-Train*)

```
>>> X=salt_data[['petal length', 'petal width','sepal length']]
    y=salt_data['species']
    X
    y
```

```
X=salt_data[['petal length', 'petal width','sepal length']]
```

```
y=salt_data['species']
```

```
X
```

	petal length	petal width	sepal length
0	1.4	0.2	5.1
1	1.4	0.2	4.9
2	1.3	0.2	4.7
3	1.5	0.2	4.6
4	1.4	0.2	5.0
...	...	...	...
145	5.2	2.3	6.7
146	5.0	1.9	6.3
147	5.2	2.0	6.5
148	5.4	2.3	6.2
149	5.1	1.8	5.9

150 rows × 3 columns

```
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int32
```

16-Veri kümemizi test ve eğitim/öğrenme kümeleri olarak ikiye bölelim-ayıralım; (*Test-Train*)

**Verileri; 75% Test, 25% Eğitim olarak ayıralım.*

>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.75,shuffle=True, random_state=5)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.75,shuffle=True, random_state=5)
```

17-Modelimizi eğitim seti üzerinde eğitip, test seti üzerinde tahminler gerçekleştirelim;

**Verileri; 120 ağaç sayısı olarak ele alalım.*

>>> clf=RandomForestClassifier(n_estimators=120)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)

```
clf=RandomForestClassifier(n_estimators=120)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
```

18-Son olarak modelimizin ne sıklıkla doğru olduğunu kontrol edelim; (*Accuracy Score*)

>>> print("Accuracy Value:",metrics.accuracy_score(y_test, y_pred))

```
print("Accuracy Value:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy Value: 0.9557522123893806
```

Önem değeri düşük olan özellikleri/parametreleri (*Sepal Width*) analizimizden çıkardıktan sonra doğruluk değerinin arttığını görebilirsiniz.**Yanıltıcı verilerin/parametrelerin analizden çıkarılması.*

Görselleştirme işlemleri için dilerseniz **Python üzerinde ki çeşitli kütüphaneleri kullanarak** dilerseniz de geçmiş dönemlerde uygulamış olduğumuz Power BI ile görselleştirme (**1**–**2**) çalışmalarından faydalanabilirsiniz.

Günlük Hayatta Algoritma Uygulama Alanları

Finans-Bankacılık Sektörü

- Kredi ve Risk Değerlendirilmesi (Analizleri),
- Dolandırıcılık Değerlendirmeleri (Fraud)
- Opsiyon Fiyatlama Sorunu Analizleri

Sağlık Hizmetleri Sektörü

- Kısmi Biyolojik Hesaplamalar
- Gen Analizleri
- Biyobelirteç Keşfi ve Analizleri

E-Ticaret Sektörü

- Çapraz Satış Analizleri
- Müşteri Öneri Sistemleri

Avantaj Dezavantaj

Rastgele Orman algoritması avantajları olarak;

- Sınıflandırma ve Regresyon problemleri üzerinde kullanılabilmektedir,
  (*Çok yönlü esnek çözüm imkanı*)
- İyi tahminleme oluşturmanızı sağlayan güçlü hiperparametreler bulunmaktadır,
- Yüksek öneme sahip özelliklerin belirlenmesinde kolaylık sağlamaktadır,
- Aşırı Uyum/Öğrenme (*Over Fit*) problemini önlemektedir-azaltmaktadır,
- Eksik verileri (*Bilinmeyen*) işlemek için önemli bir çözümdür,
- Büyük veri kümeleri için verimli işleme ve sonuç üretmektedir,
- Kural tabanlı bir yaklaşım kullandığı için verilerin normalleştirilmesi gerekli olmamaktadır,
- Paralelleştirilebilir-Paralel işlemler gerçekleştirilebilmektedir.
  (*Analiz ve işlem sürecinizi çalıştırmak için birden fazla makineye bölebilirsiniz.* ***n_jobs***)

Rastgele Orman algoritması dezavantajları olarak;

- Yapısal olarak oluşturmuş olduğu fazlaca çözüm (*Ağaç*) sayısından dolayı biraz yavaş sonuç üretmektedir, (*Eğitim hızlı, tahmin uzun*)
  (*Özellikle gerçek zamanlı tahminlemeler için*)
- Daha doğru bir tahminleme için daha yavaş sonuçlanan bir modelle yüksek sayıda çözüm (*Ağaç*) gerekmektedir,
- Verilerin ekstrapolasyonunda* ideal değildir,
  **Ekstrapolasyon; bilinen veri noktalarının ayrık kümesi dışında yeni veri noktaları oluşturma işlemidir.*
- Veriler çok seyrek olduğunda iyi sonuçlar üretmemektedir,
  (*Özelliklerin alt kümesi ve önyüklenen örnek değişmez bir uzay üretmektedir.*)
- Tanımlayıcı bir araç değil, tahmine dayalı bir modelleme aracıdır.
  (*Yani; verilerinizdeki ilişkilerin bir tanımını arıyorsanız, diğer yaklaşımlara yönelmeniz daha ivi olacaktır.*)