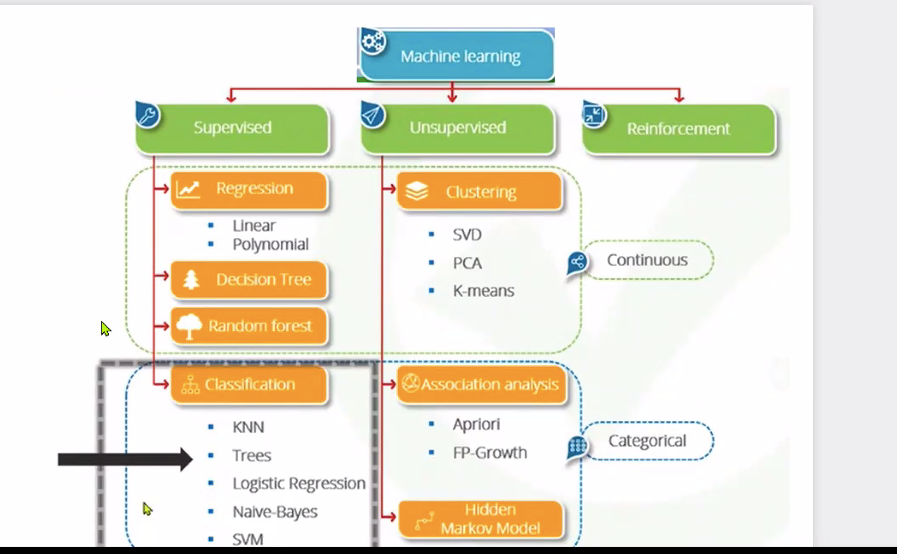
**ML\_Decision Tree\_28\_07\_2023**



If else ile calisan bir algoritma decision tree

classification ile regression amaçlı kullanılıyor.

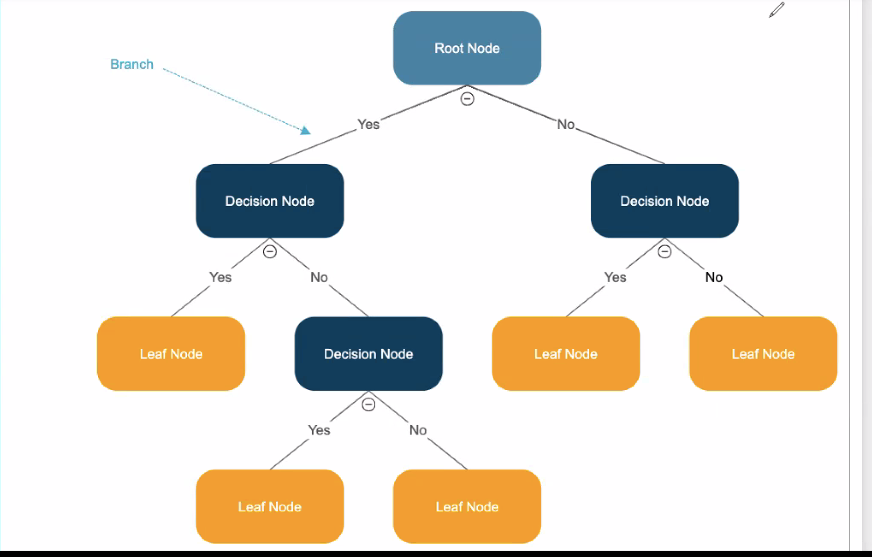
Continous datalarda--> regression olarak kullaniliyor

Simdi ise classification amacli kullanacagiz.

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import DecisionTreeRegressor

decision tree için iki ayrı modeli bu şekilde çağırıyoruz



Koku yukarida bir agac dusunun.

Gini katsayisi en dusuk olacak feature i aliyor root node atiyor.

Sonrasindaki decision node’lar ise ozellikleri iceren feature’lar. Bunlara yonelik nodelar oluyor. Ilk node dan bolundukten sonraki node’lar. Bolunmenin devam ettigi yerler--> **decision node**

**Leaf node** ise kararin verildigi nokta.

Agacin sonunda karar verildigi nokta--> leaf node. Verilen karar burda yazar. Sinifi belirledigi yer bu. Yanlis da olsa burda belirliyor olacak.

Mesela 1 ve 0’dan olusan binary class bir datasetim olsun.

X 1’den buyuk mu--> hayir--> 6 degeri bu sekilde biliyor basta. Pure bir yesil class, bunu belirliyor. Leaf node.

X2 1’den buyuk mu sorusuna gore de 2 class ayirdi:

**X2 1’den buyuk ise, 2 class var:**

-1.8’den buyuk ise sag ust kosedeki yesillerden olusan pure bir class elde ediyor. Leaf node.

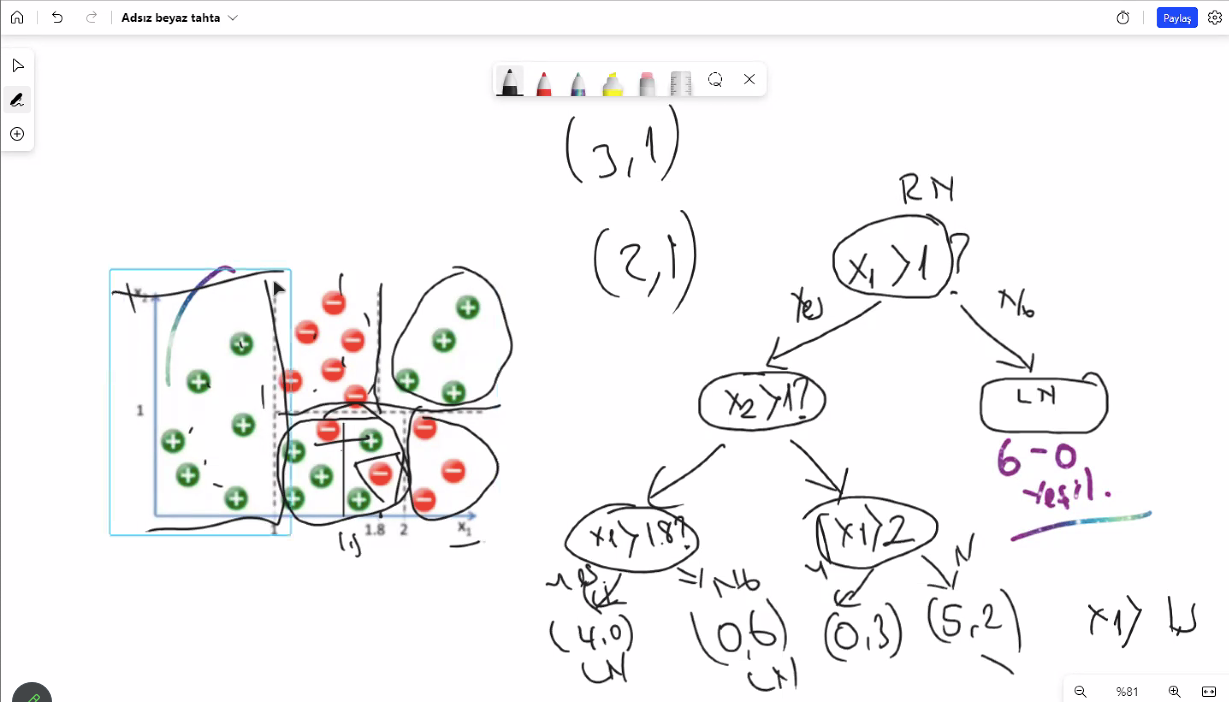
-1.8’den kucuk ise ust ortada kalan kirmizilardan olusan pure bir class elde ediyor. Leaf node

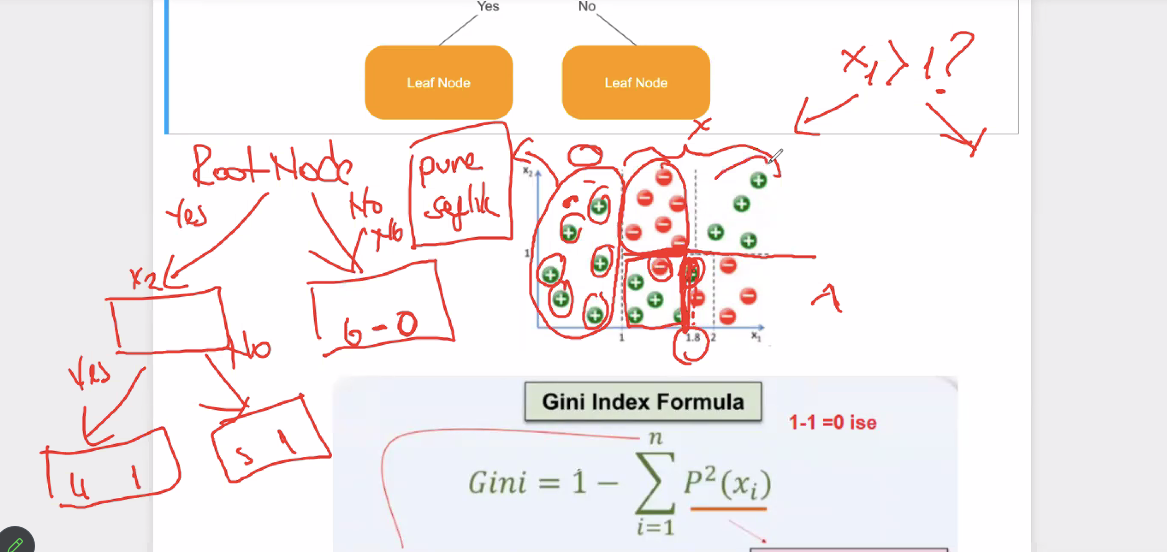
**X2 1’den kucuk ise:**

Altta kalan kisim icin, bu sefer

X1 2’den buyuk mu diye sorunca, evet ise en sagdaki hepsi kirmizi olan classi da pure olarak belirlemis oluyor. Leaf node

**X1 2’den kucuk mu diye sorunca, evet ise**-->alt ortadaki kirmizi ve yesil karisik olan class icin yeni sorular sormali. Ama bir yerde durdurmazsam, hepsini bilinceye kadar sorar. Ama sonunda overfitting olur.





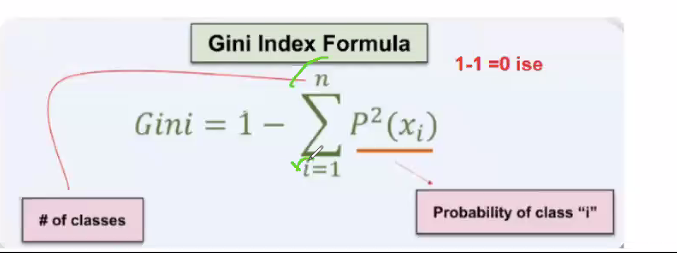
Saflik kalana kadar devam eder bu isleme. Tum hepsi kendi sinifina ayrilana kadar.

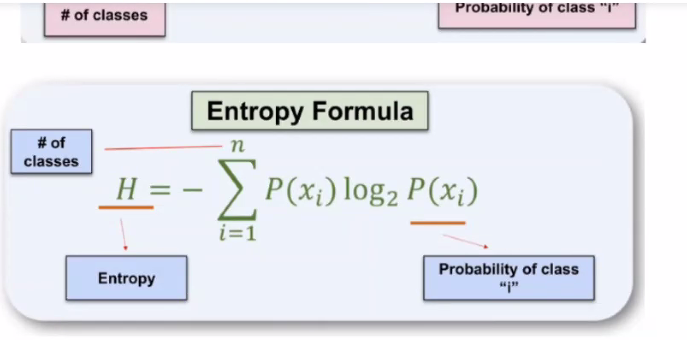
Ama tum siniflar birbirinden ayrildigi zaman da overfitting e gider. Yani Agacin bolunme sayisini sinirlamazsam, sonunda overfitting olur. Decision tree’nin handikapi da bu.

Bunu onlemek icin max\_depth parametremiz var. max\_depth= Karar Ağacının maksimum derinliğini ifade eder. Değer girilmezse limitsiz olur. Model overfit(aşırı uyum) olmuşsa düşürülmesi gerekir.

Arkasindaki Calisma prensibi ise--> 2 tane katsayi var. Gini ve entrophie

Information gain de denir. Ama asil entrophie ile hesaplanan information gain kastedilir. Asil kullanilan ise gini’dir.

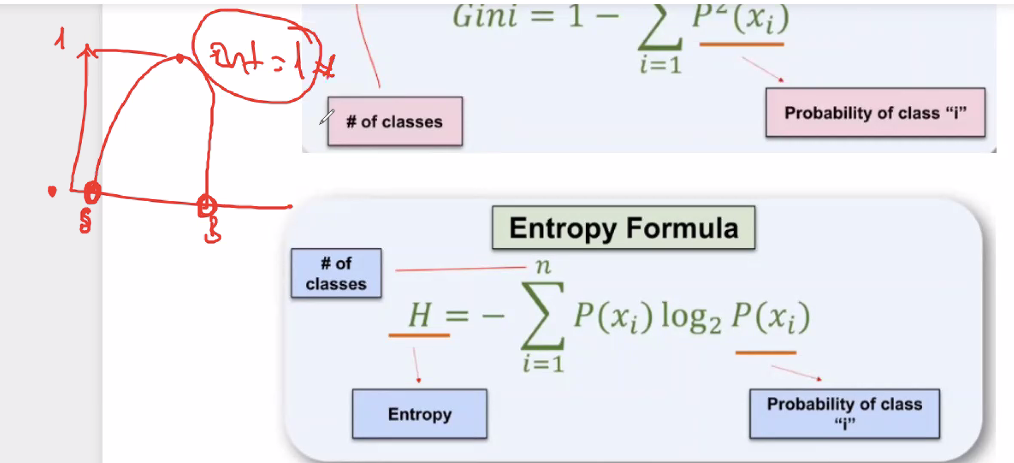




Bir sinifin olasilik hesabini yapar, 1’den cikartir.

Gini eger 0 ise--> purity’den bahsederiz. Gini’nin dusuk olmasi istedigimiz bir sey

Mesela en bastaki klass’ta yesil gelme olasiligi 1 oldugu icin gini 0’dir.



Entrophy 1 ise maximum deger en kaotik degeri entrophy’nin. 0 purity’i temsil ediyor.

1 karmasikligi temsil ediyor entrophy’de.

Logoritmik hesaplamalarda biraz daha karmasik entrophy.

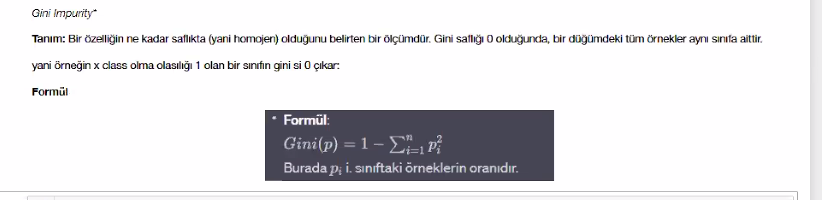
Ama ginide 0.5 karmasikligi temsil ediyor,

Ancak entrophy daha dengeli agaclar verebiliyor. Daha karmasiksa daha iyi skorlar verebiliyor.Bir sinif fazla ise gini’nin ona kayma riski olabiliyor.

Gini daha hizli oldugu icin tercih ediliyor genelde. Biz de genelde gini tercih edecegiz. Islem maaliyeti acisindan daha avantajli. Mumkunse ikisine de bakip kiyasla.



**Gini impurity**



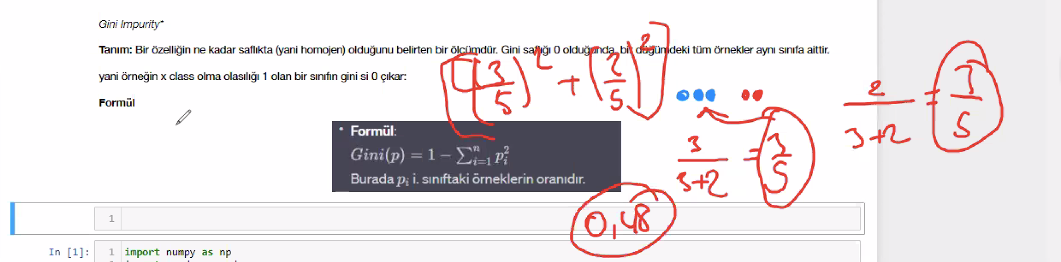
3 mavi 2 kirmizi top olsun bir root node’da.

Bu durumda **gini impurity** nedir?-->

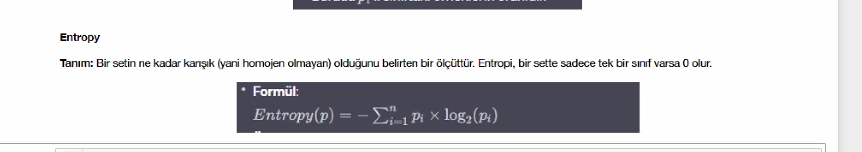
Mavi olma olasiligi=3/5

Kirmizi olma olasiligi=2/5

Ilgili formule : gini katsayim 0.48 cikar.



**Entrophy:**

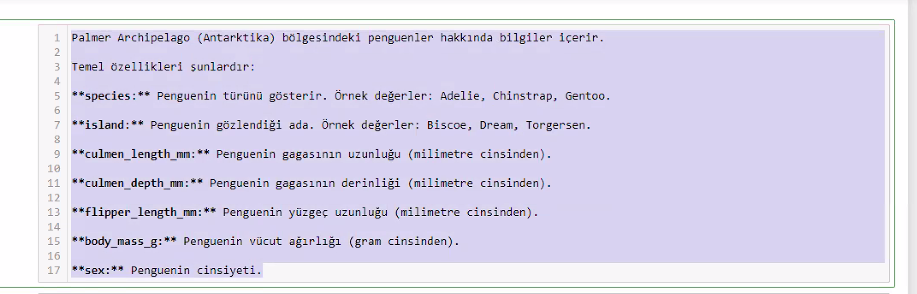


Ayni ornek icin hesaplaninca 1,7 olarak cikar entrophy.

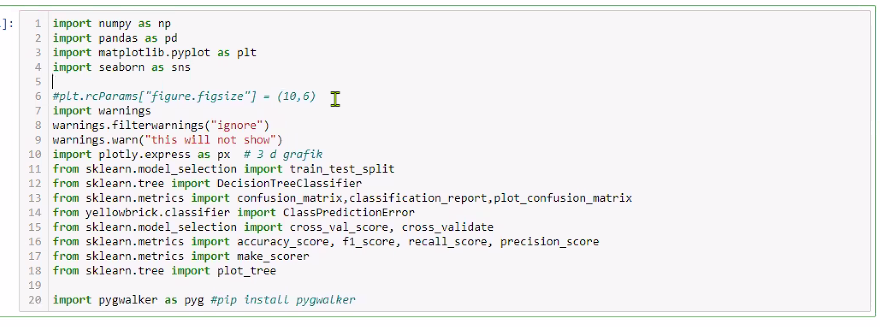
Ozellikle giniyi tercih edecegiz.

Hepsi arti ise mesela 0 cikar

2 arti 2 eksi var ise 1 cikar.



Uzunluk-derinlik mesela turleri etkilemede etkili bir faktormus



Decision tree multicolinearity’e karsi dayaniklidir. Multicolinearity sorunundan etkilenmez

Eğerki 0.87 olan değer 0.90 üzerinde olup multicolinearty sorunu olsa idi

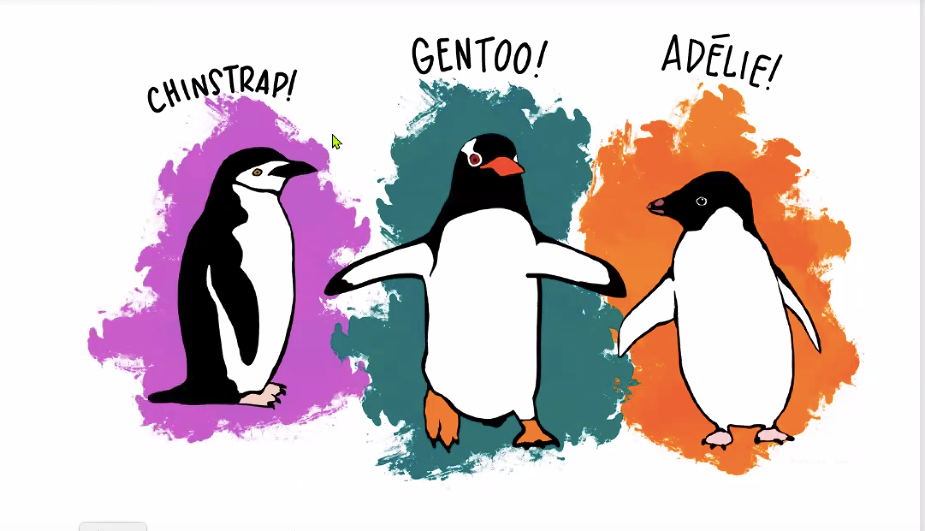
Karar ağaçları çoklu doğrusal bağlantı (multicollinearity) problemlerine karşı dayanıklıdır çünkü bu modeller doğrusal olmadığı için featureler arasındaki doğrusal ilişkileri dikkate almazlar. Yani, bir karar ağacı modeli eğitirken multicollinearity'den kaynaklanan bir sorun yaşanmaz.

**Get\_dummies**

drop\_first=True fonksiyonu

|  |
| --- |
| **Get dummies (Marcus hoca recap)**  **Unique degerleri birbirinden ayirmak icin bir kodumuz var.**  Air conditioning, cruise control, climate gibi cesitli fonksiyonlar var. Bunlari ayirabiliriz mesela. Bunlari ayiriyoruz bu kodla. Sonra bunlara get\_dummies yapiyoruz. Ama bizim dummy tuzagi diye bir sorunumuz vardi. Bunun icin drop first yapiyoruz ornegin. Boylece de model complexity’i azaltmis oluyoruz. Yoksa fazla feature ile model iyilesmis gibi de gorunebilir ama overfittinge neden olan yalanci bir iyilesmedir bu aslinda. Ondan adi dummy tuzagi.  3 feature var diyelim, aracimiz LPG diyelim LPg’ye 1 yaziyor digerlerine 0 yaziyor  Diesel olsa diesel’e 1 yazacak digerlerine 0 yazacak. Digeri otomatik olarak kodlanmis olacak, yani benzinli. 3 tane ise 2 tanesini kodlamam yeterli  Elimizde 2 tane unique deger varsa, benzine 1, gaza 0 verecek. Bir kere kodlamam yeterli.  Get\_dummies yapinca artik object sutunumuz yok, hepsi artik numeric. Modele hazir.  #Ordinallik varsa Get\_dummies yapinca modeli yaniltir. Mesela aralarinda bir siralama var, pahali, ucuz, orta gibi  ordinal encoder kullanacagiz ordinal bir data varsa.  Arada ordinallik varsa, ona gore bir encoding kullanacagiz.  One\_hot encoder mesela yine get\_dummies yontemi-->hepsine ayni onemi veriyor. |

Get\_dummies yapinca data leakage olmaz.



**Eksik datayi doldurma:**

yukaridaki tabloyu ve eksik veri olan satiri karsilastirinca culmen\_depth\_mm 15.7 oldugu ve

# culmen\_depth\_mm Female de max 15.5 olabildigi goruluyor bu sebeple eksik cinsiyetin MALE olduguna karar veriyoruz

Nokta tek bir gozlem olarak var. Missing value ile mucadele ornegi bu.

Gentoo Culmen depth--> 15.7 max imis

Bu penguen male olabilir diye dusundum.

**Barplotla bakinca**

Datamin unbalanced data oldugu sinyallerini aliyorum.

**Pairplotla bakinca**

Griftlik durumuna bakip yorumluyorum. Flipper\_length’de ayrisma daha iyi diye anliyorum.

**Pgwalker**

Tableu’nun ML’de embed hali gibi dusun. Degiskenleri atip tablolarini cizdiriyorum

Species secip color kismina atarasan cinsiyete gore renklendirir.

**Multicolinearity**

Decision tree’de en guclu feature’a gore bir tercihte bulunur. Bir kalssifikasyon moldei oldugu icin multicolinearity hic sorun olmuyor.

En onemli feature’i secip multicolinearity’den etkilenmiyor ama bu multicolinearity sorunu yok ya da cozuldu demek degildir. Sorundan etkilenmiyor sadece.

Yani olay multicolinearity’i cozmek icin decision tree’de bir sey yapmama gerek yok.

**Feature Engineering**

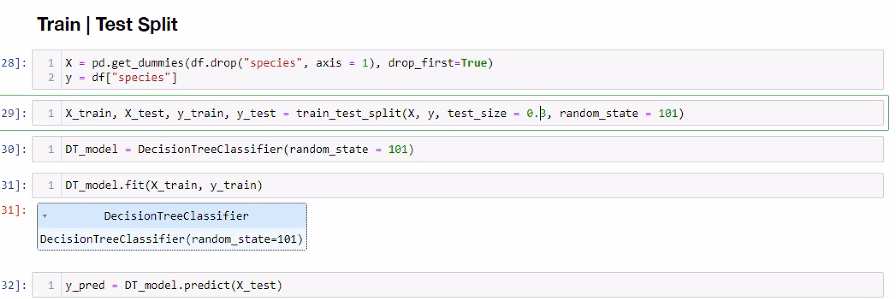
Get\_dummies

Get\_dummies String olanlari binary numeric olanlara ceviriyor.

3 class varsa 3 feature haline getiriyor. Aralarinda ordinallik olmadigi icin get dummies kullaniyoruz. Get\_dummies sayinin degerini goz ardi eder.

Island feature’unda 3 class var--> get\_dummies yapinca 2 tane fazladan feature’um oldu, 0-1 seklinde atama yapti. Ucuncu island cinsi drop=first dedigimi icin ayri bir sutunda ayri bir feature olarak gorunmuyor ama iki tip island’in da 0 oldugu satirda 3. adayi kastettigini anliyor model.

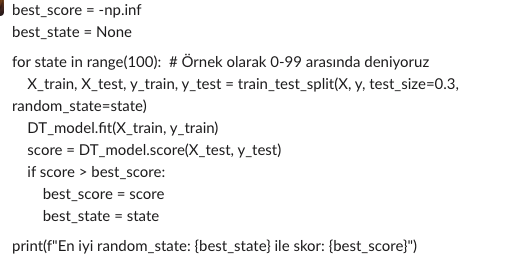
Bu arada kategorik data targetta ise, ona get\_dummies yapmamaiza gerek yok. Species zaten stringdi. O sekilde kaldi. Adele ve gentoo gibi. Targetin bu sekilde olmasina musaade ediyor.

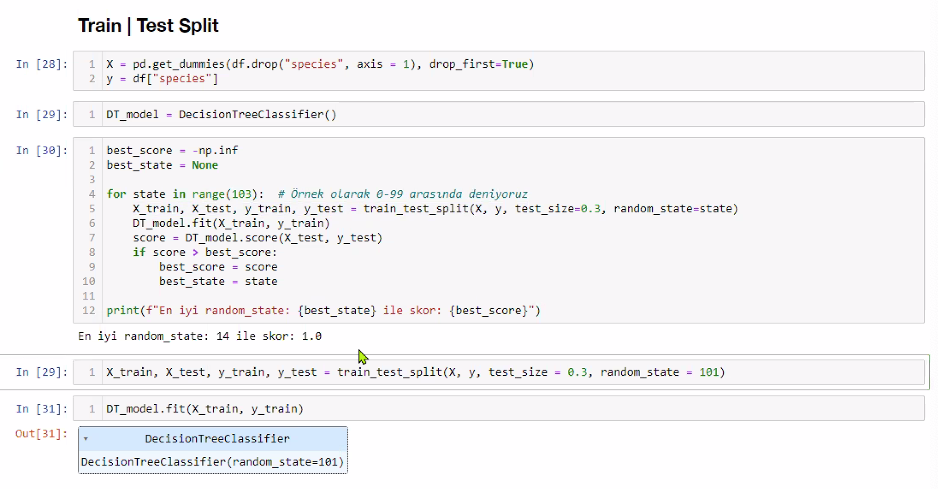


test\_size-->0.30 verdik.

random\_state-->101

best\_score for random state:





Random\_state 14 ile en iyi skoru aliyormusum. Ama 101 ile devam edecegim.

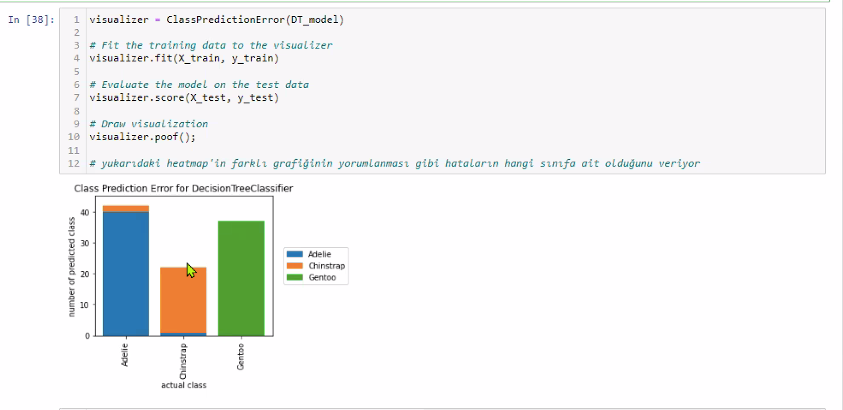


# For default settings of figures;

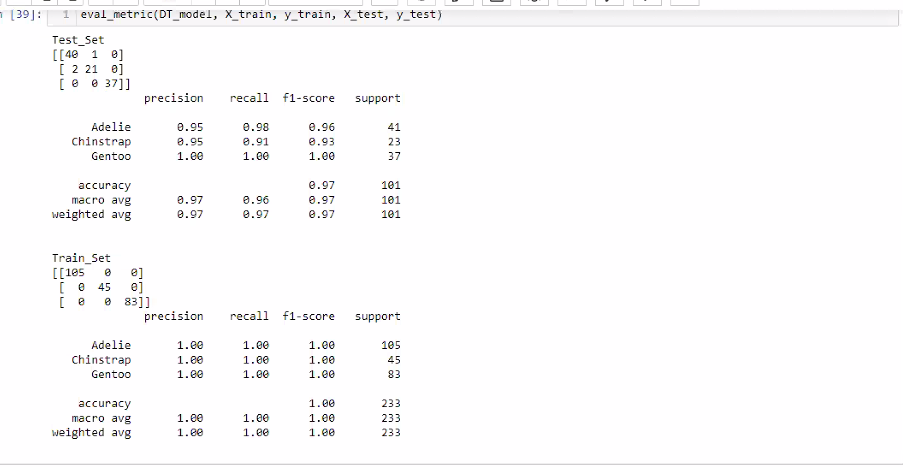
plt.rcParams.update(plt.rcParamsDefault)

(bu kodu calistirinca sorunlar duzeliyor, default degerlere donuyormus)

Diagonal olarak test datasinda dogru bildiklerini gosteriyor.



2 tane classta hatamiz var su an.



Confusion matrix geldi ve klassifikasyon

Trainde modelin basarisi %100

Testte centoda basari %100

Adel’deki basari biraz daha iyi chinstrap’a gore.

Overfitting var denemez.

Chinstrap’ta fark biraz daha fazla

Multiclass oldugunda macro avg ve weighted avg degerlerine bakacagim.

Accuracy’de 1 deger var. Mikro skor accuracy ile ayni oldugundan bunun accuracy olarak goruruz. Accuracy’de aldigi deger, precision, recall ve f1’de ayni oldugu icin tek accuracy getirdi.

**#Eger target datam balanced ise, yani target’daki sinif sayilarim esit veya yakinsa balanced data--> accuracy yani micro avg**

**#Datamda dengesizlik var ise, unbalanced data--> macro avg**

**#eğer datam dengesiz/dengeli ancak ben sayısı yüksek olana göre bakmak istiyorsam--> weight avg**

Continous datalardan 1 skorunu yakalamak kolay degildir. Train’de 1 yakalamak mumkun degildir. 1’i gorursen overfitting’den suphelenmen lazim.

Klassifikasyon datalarinda ise 1 gorme ihtimali daha yuksek. O yuzden hemen overfitting dememek lazim.

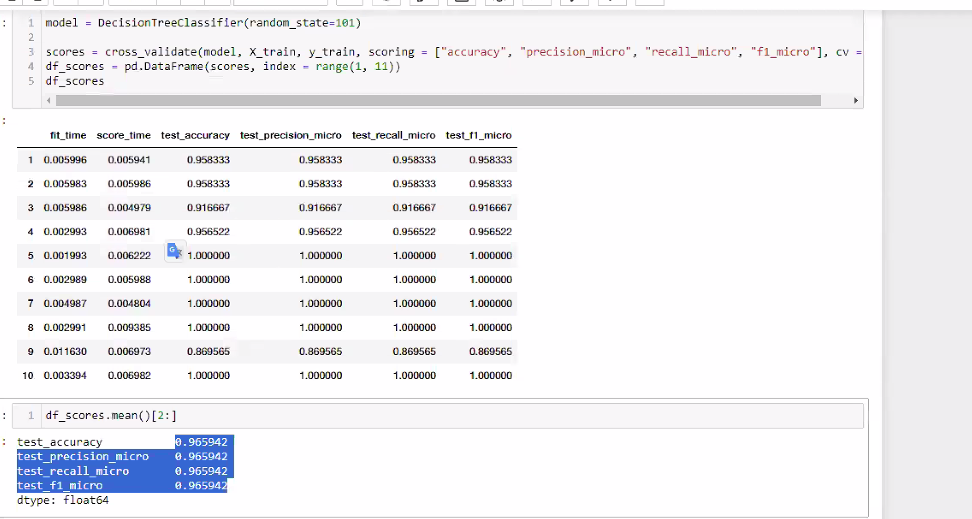
Decision tree’de en buyuk handikap cabuk overfitting’e dusmesi.

Bunun onune gececegimiz bir parametre var.

Yani ilkel skor iyi geldi.

Ama train\_setin hepsinin 1 gelmesi suphe uyandiriyor. Yine de test cok yuksek oldugu icin uyum var diyebiliriz.

**Cross Validation**

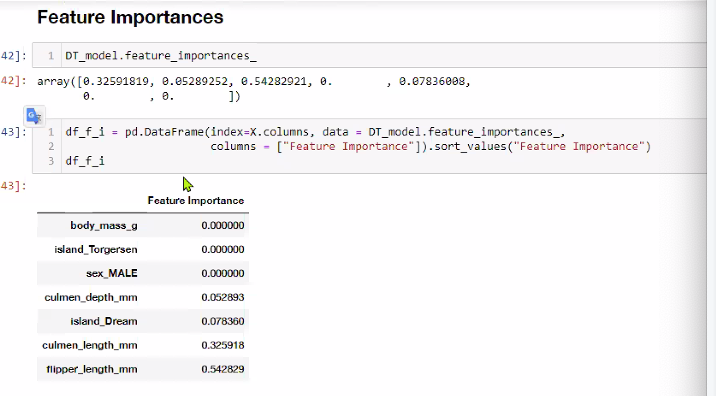


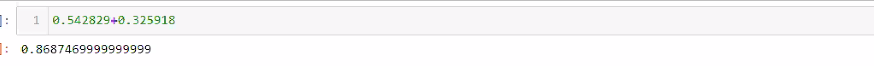
Olmasi gerekn bu degil ama accuracy ile mikro degerlerinin ayni oldugunu gostermek icin bunu yaptim.

Hepsi 0.9659

Yani micro skorum ile accuracy ayni anlama geliyor.

Recall, precision ve f1 de dolasiyla. Hepsi ayni





Bu en son cikan 2 feature’un toplami. Hepsini toplayinca 0.99 oluyor.

Feature importance deyince biz lasso’yu kullaniyorduk. Once scale yapiyorduk hatta.

Decisiontree’de feature\_importance fonksiyonu ile dogrudan yapabiliyoruz.

3 tane feature’a 0 vermis, lasso gibi davranmis.

DT onemli bir feature bulursa, ondan devam eder. Digerlerini modele koymaz, lasso gibi davranir.

(Random forest ile yapacagiz feature importance’i artik.)

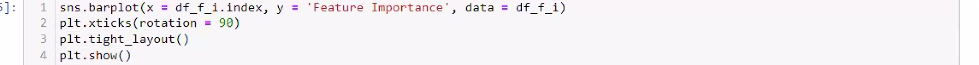
Hatta decision tree’de scale yapmadigim icin pipeline da kullanmiyorum.

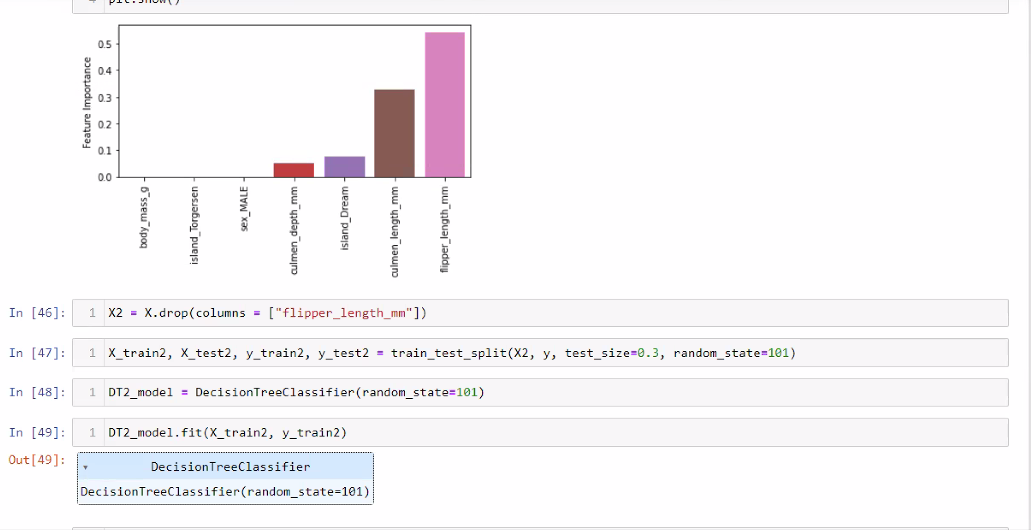
Tree based modellerde, feature importance’ina guveniriz random-forestin!

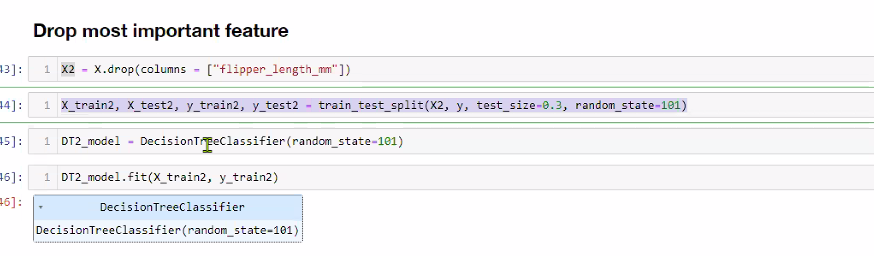
Random\_forest daha gelismis bir algoritma.decisiontree bir agacken o bir orman :)

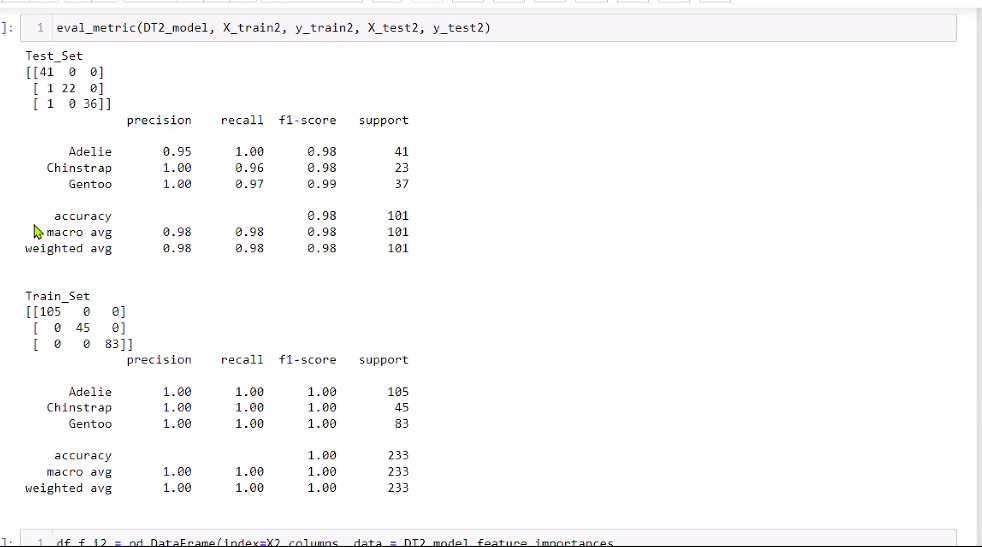
Flipper\_length’i yani en onemli feature’u bir dusurun oyle bir calistirin, o olmadan belki digerlerinden daha fazla insight alacak. Flipper\_length varken digerlerine 0 verdi gecti belki. Sonra bunlari karsilastirin.

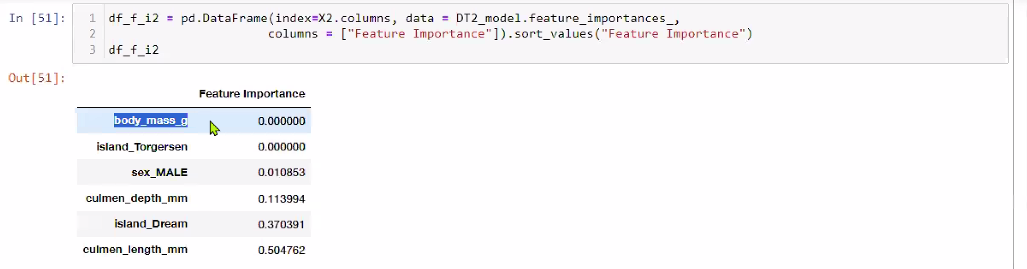
|  |
| --- |
| DT de eğer bir feature çok yüksek bir importance değerine sahipse onu düşürüp birde öyle skor alın sonra bunları karşılaştırın. Sebebi DT güçlü bulduğu featureler üzerinden ilerler diğerlerini kale almaz.  En önemlisi kanat uzunluğu ve gaga uzunluğu en önemli %90 oranında bizim için önemli  işte bu durum bizim modelimizin overfitting e gitmesine neden olabiliyor. DT önemli featurelerle eğer datanın örnke verecek olursak %90 ını ayırt edebiliyorsa diğer featureleri kullanmaz önemsizleştirir. Buda o featurelerden alacağımız insight lardan bizi yoksun bırakır.  hatırlarsanız modelimiz çok compexity olduğunda bizi overfittinge götürüyor demiştim. işte size bir overfitting sebebi daha eğer çok önemli bir feature varsa bu da bizi overfittinge götürebilir. Bunu göreceğiz. |











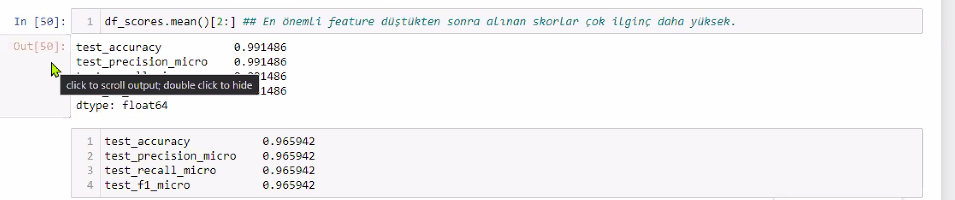
En onemli feature dusurdukten sonra gelen skorlar ilginc bir sekilde duzelmis.

Macro avg. 0.98’e cikmis.

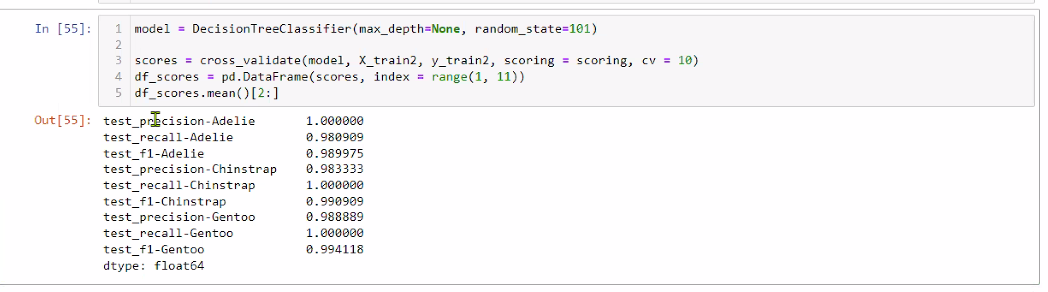
Digerlerinden daha fazla insight olan feature’i dusurunce digerlerinden daha fazla insight alabilmis.

En guclu feature’dan baslar bolunmeye yani root node olarak da onu secer.

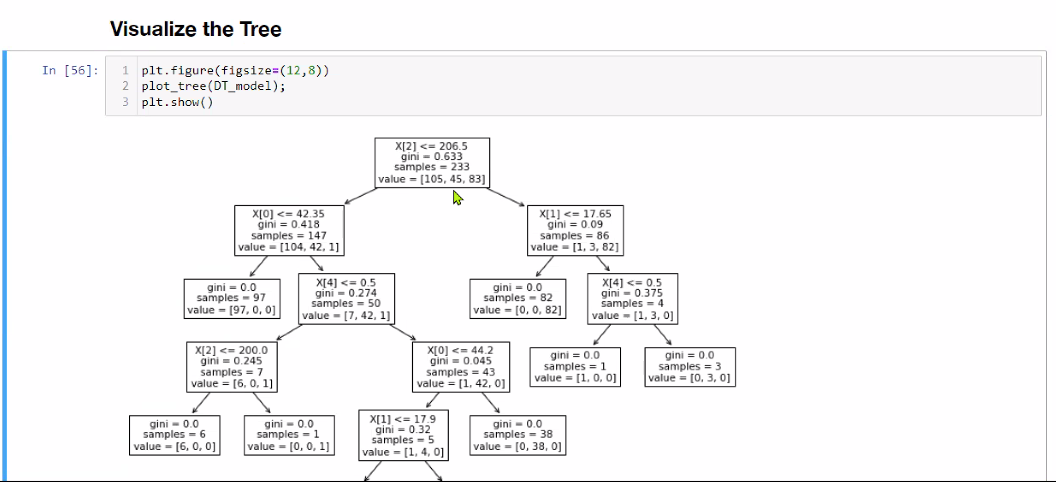
Ama drop yapinca daha da yukseldi skorlarim.



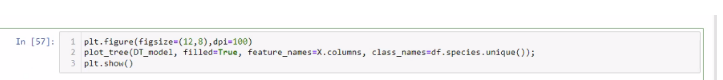


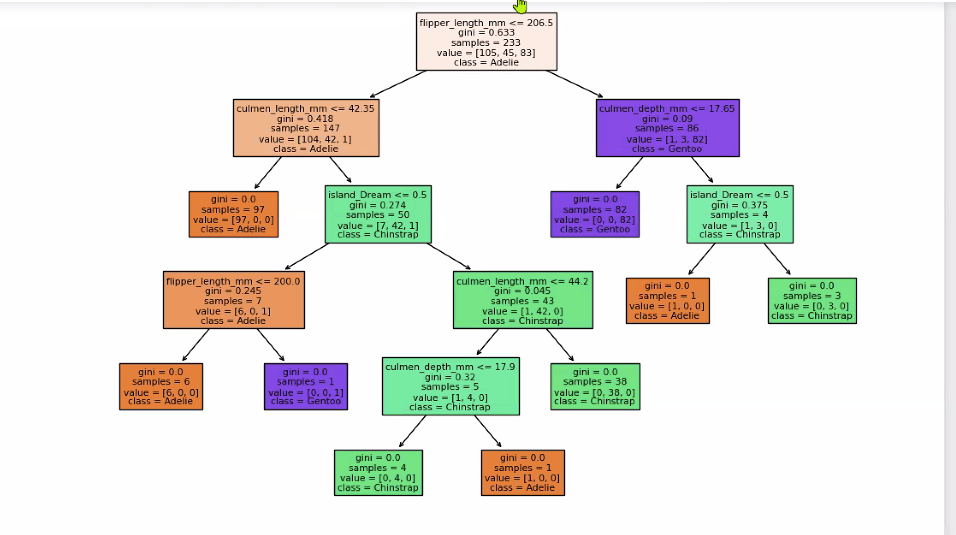


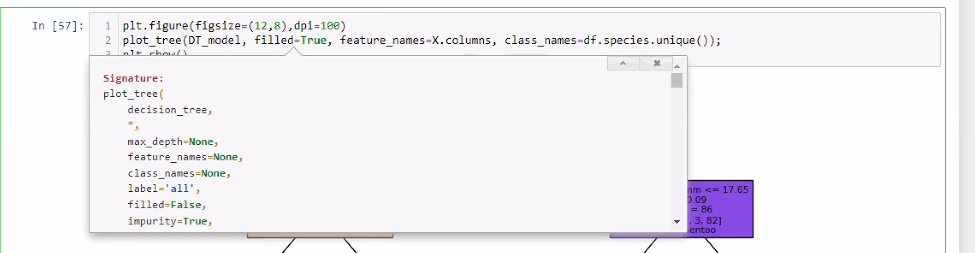
Hepsinin skorlarini aldim. Precision, recall ve f1’i hepsinin test sonuclarina ait olan skorlarini goruyorum. Birbiri ile cok uyumlu oldugu gorunuyor.



Bu x2 neyi temsil ediyor. Yazsin istiyorum. O yuzden asagidaki kodu yazdirarak daha anlasilir ve renkli bir gorsel yaptirmak icin:







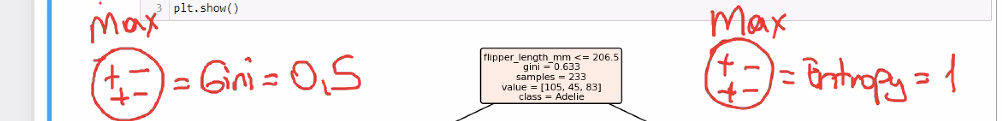
**En bastaki root node flipper length’i yorumlayalim:**

Ilk basta flipper\_length secmis. Root node. 206.5’dan kucuk olanlari sola, hayir dediklerini saga ayirmis. Sonra 2. soruya gececek.

Ama daha ilk soruda 105 adele, 45 digeri, 85 gentto diye ayirt edebilmis. Gini=0.633 cikmis, yani daha safliga epey var. Cunku gini 0 da pure oluyor. 0.5 en karmasik nokta. 0.633 ile 0.4 gibi oranlar ayni karmasikligi temsil eder. 0.5 en buyuk risk. 0.5’in alti ve ustu es deger sekilde karmasiklasiyor. Yani 0.633 olmasi 0.5’ten daha iyi bir durum.

Gini 0 olunca pure oluyor. Ama teorik olarak 1 olmasi mumkun degil. 1’e yaklastikca setteki ogelerin rastgele yaklastigini soyleyebiliriz.

(ama entrophy’de 1 olmasi cok kotu. Target’da 3 class olmasi 1’in uzerinde cikmasina neden olabiliyor entrophy’nin).



Peki flipper\_length’deki 206.5 rakamini nasil belirliyor sormak icin?

Bu en iyi nokta imis,105 adele, 45 digeri, 83 gentoo diye ayirt edebilmis.

**Icinde train datasi var.**

**Filled\_True-**-> butun bosluklari doldur demek

**class\_names**--Class’lardaki unique isimlerine gore isimlendir demek

**Max-depth--**> her birinin leaf node ile bitmemesi, mdelin overfittinge gitmemesi icin kullanacagim parametre. En cok bunu kullanacagiz. random forest hep kisa tutar.

**Impurity\_True**--> safik gosterir. Burda gini gosteriyor.

**Precision**--> noktadan sonra basamak sayisi

|  |
| --- |
| decision\_tree: Görselleştirilmek istenen karar ağacı nesnesi.  max\_depth: Ağaçtaki maksimum derinlik. Eğer belirtilmezse tüm ağaç gösterilir.  feature\_names: Özelliklerin (features) isimleri. Bu sayede ağaç üzerinde gösterilen özelliklerin daha anlamlı isimleri olabilir.  class\_names: Hedef sınıfların (target classes) isimleri. Bu, ağacın sonunda veri noktalarının hangi sınıflara ait olduğunu göstermek için kullanılır.  label: 'all', 'root', 'none' arasında bir değer olabilir. Hangi düğüm etiketlerinin gösterilip gösterilmeyeceğini belirtir.  filled: Boolean değeri. True olduğunda, düğümler sınıflara göre renklendirilir.  impurity: Boolean değeri. True olduğunda, düğümlerde saflık (genellikle Gini saflığı) gösterilir.  node\_ids: Boolean değeri. True olduğunda, her düğüm için düğüm ID'leri gösterilir.  proportion: Boolean değeri. True olduğunda, her düğümdeki örneklerin oranı gösterilir.  rounded: Boolean değeri. True olduğunda, düğüm kutularının köşeleri yuvarlanır.  precision: Noktadan sonraki basamak sayısı. Bu, gösterilen ondalıklı değerlerin hassasiyetini ayarlar.  ax: Matplotlib eksen nesnesi. Eğer belirtilmezse mevcut eksen kullanılır veya yeni bir eksen oluşturulur.  fontsize: Yazı tipi boyutu. Düğümlerdeki metnin boyutunu belirtmek için kullanılır. |

**Understanding hyperparameters**

criterion: Karar ağacını bölerken kullanılan fonksiyonu belirtir. 'gini' (Gini saflığı) veya 'entropy' (bilgi kazancı) olabilir.

splitter: Her düğümdeki bölünmeyi nasıl seçeceğini belirtir. 'best' en iyi bölünmeyi seçerken, 'random' rastgele bir bölünme seçer.

max\_depth: Ağacın maksimum derinliği. Ağacın ne kadar derin olabileceğini sınırlamak için kullanılır.

min\_samples\_split: Bir düğümü bölmek için gereken minimum örnek sayısı.

min\_samples\_leaf: Bir yaprağın (düğümün sonu) sahip olması gereken minimum örnek sayısı.

min\_weight\_fraction\_leaf: Bir yaprakta bulunması gereken minimum ağırlıklı örnek oranı.

max\_features: Her bölünmede dikkate alınacak maksimum özellik sayısı. 'auto', 'sqrt', 'log2' veya bir tam sayı olabilir.

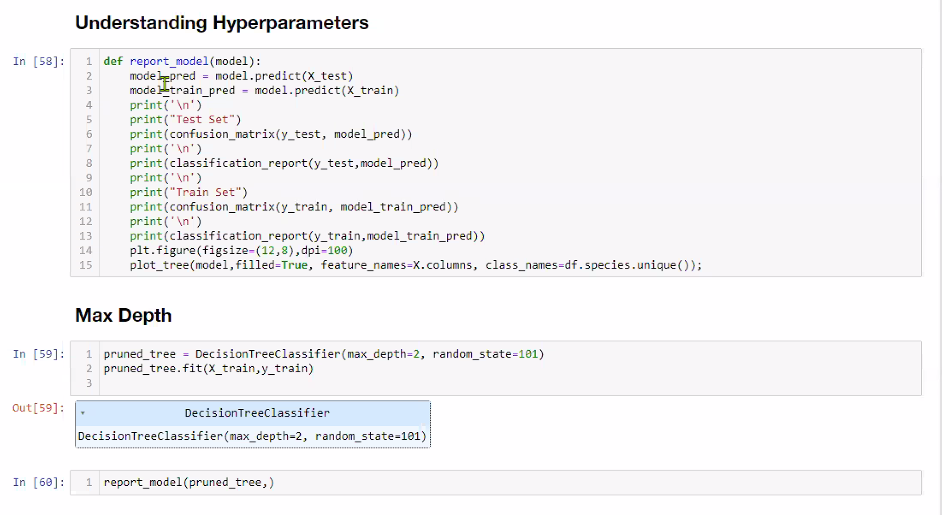
random\_state: Rastlantısallığın kontrol edildiği parametre. Modeli yeniden çalıştırdığınızda aynı sonuçları elde etmek için kullanılır.

max\_leaf\_nodes: Maksimum yaprak düğümü sayısı. Bu sayıyı sınırlayarak ağacın büyümesini sınırlayabilirsiniz.

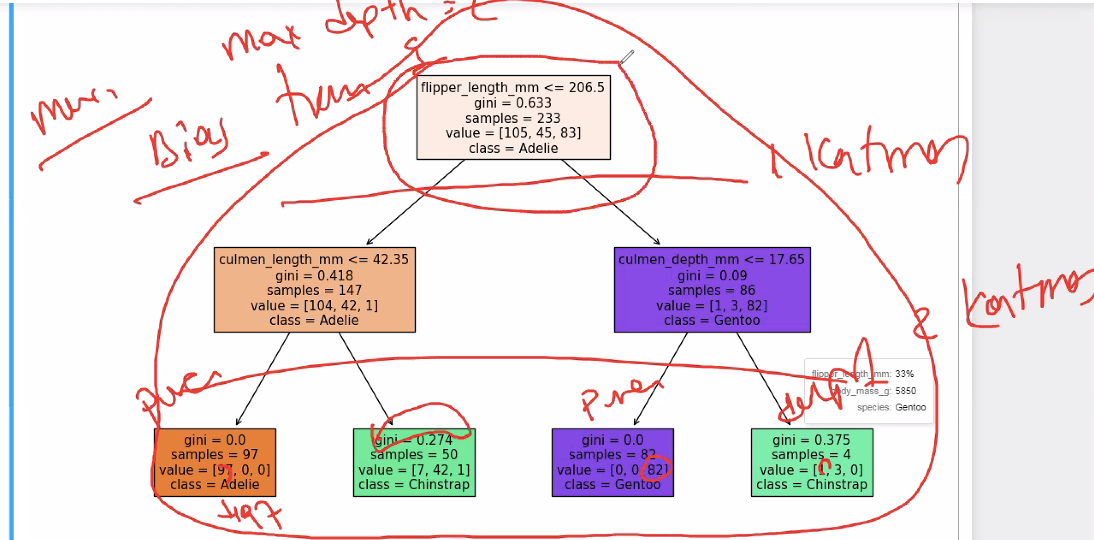
min\_impurity\_decrease: Bir düğümü bölmek için gerekli minimum saflık azalması.

class\_weight: Sınıfların ağırlıkları. Dengelemeyi veya özel sınıf ağırlıklandırmasını yapmak için kullanılır.

ccp\_alpha: Cost-complexity pruning'in (Maliyet-karmaşıklık budaması) alfası. Ağacın budamasını kontrol etmek için kullanılır. Daha yüksek değerler, daha fazla budamaya neden olur.



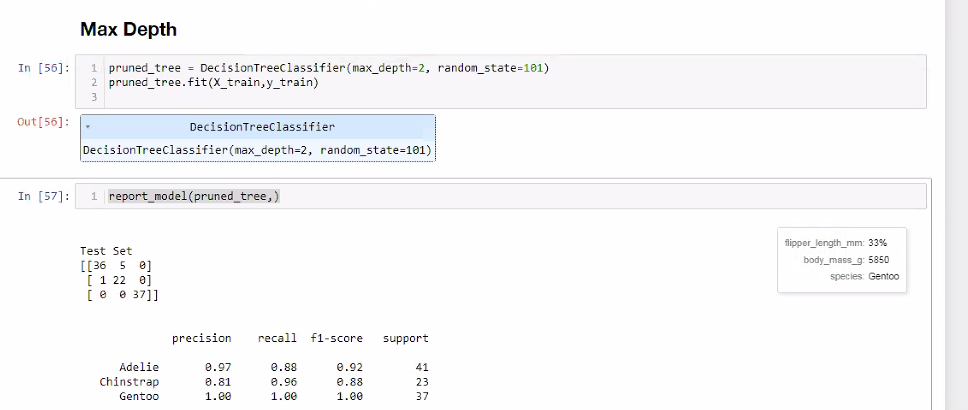
Eval\_metric’te aldigimiz skorlarin aynisi.



Max\_depth=2 girdigim icin modelim 2 katmanda hemen sonuca ulasmaya calsiir ama hatalara yer verir modelim. Burda mesela Gini katsayilari purity’den uzak kalmis.

Max\_depth ile modelime bias ekliyorum

Train’deki skorlari dusururken test skorlarimi yukseltmis oluyorum.





Overfitting’i onlemek icin en iyi parametremiz max\_depth

2-3-4-5 deneyin. Gridsearch icin de kullanabilirsin bunu

Bundan sonrakiler ince ayarlar, temel max\_depth

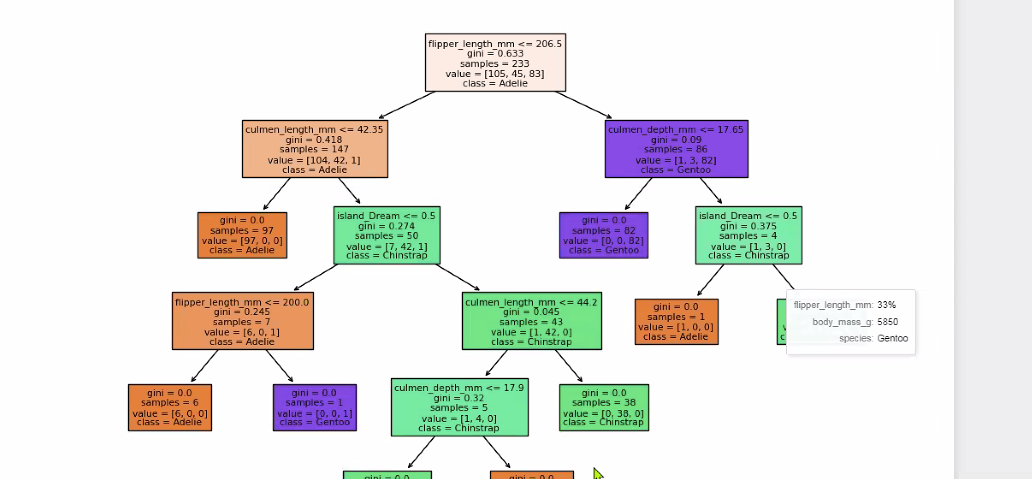
**Splitter=best**

Random secersem besti bulamadigi icin agac dallari artar

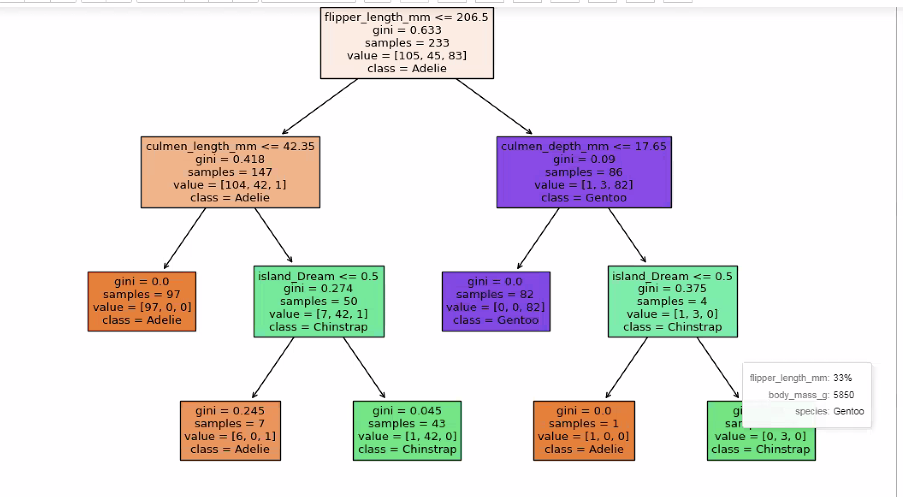
Ama besti secersem oyle olmaz

min\_samples\_leaf--> yaprak olarak kabul edilmesi icin gozlem sayisi

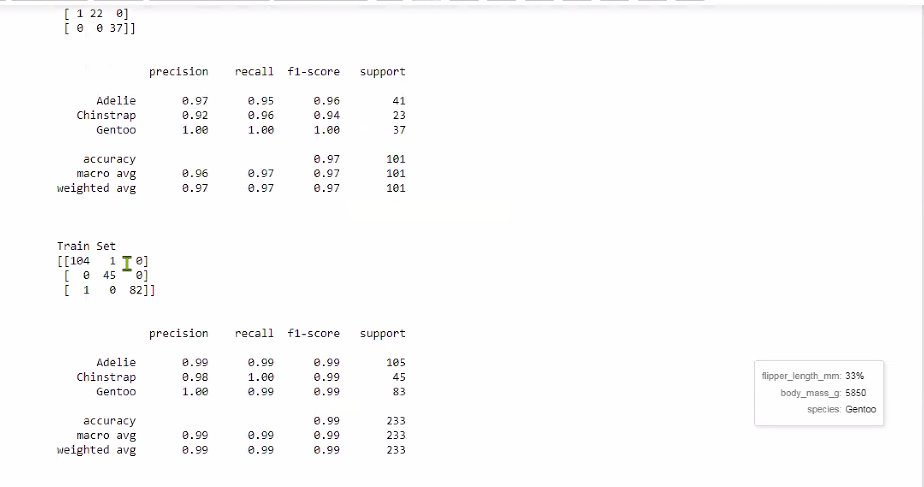
**Max-depth 5 verdim:**



**3 verdim:**

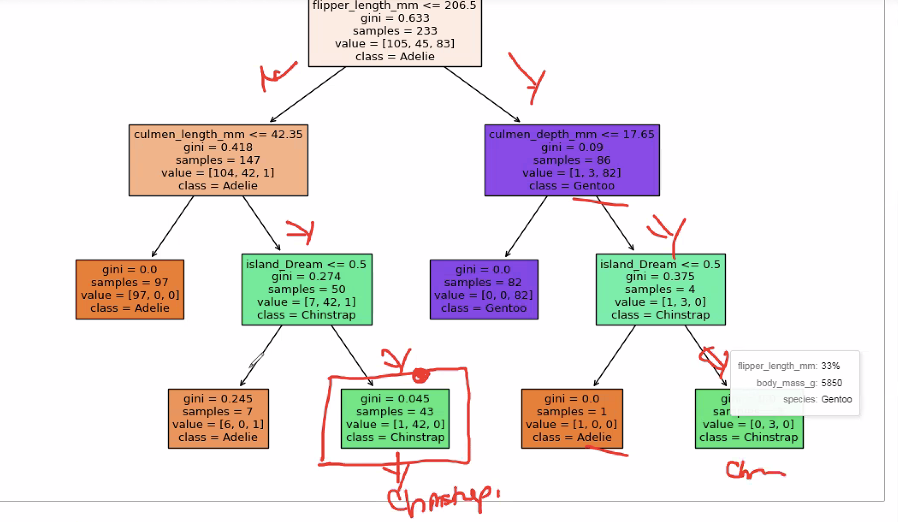


3 verince eval metric



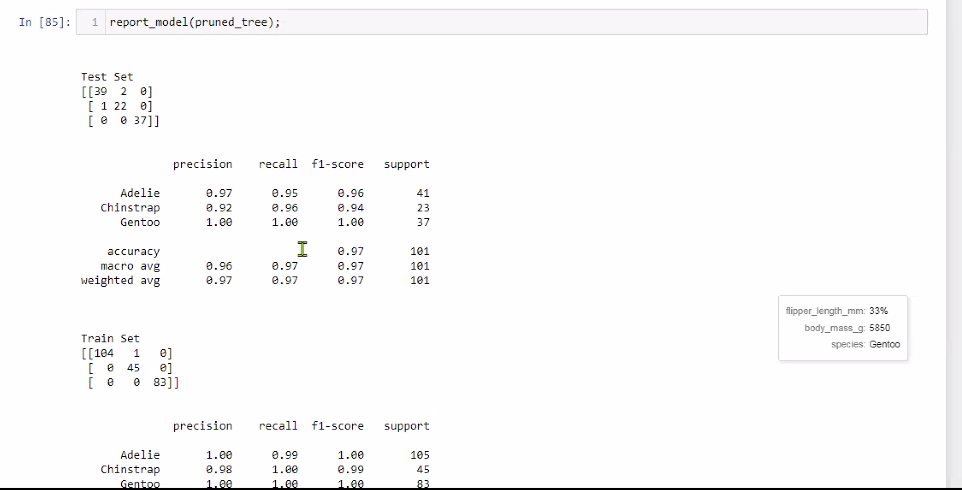
Biraz bozulmus 5’e gore skorlarim.

**Tahmini nasil yapiyor:**

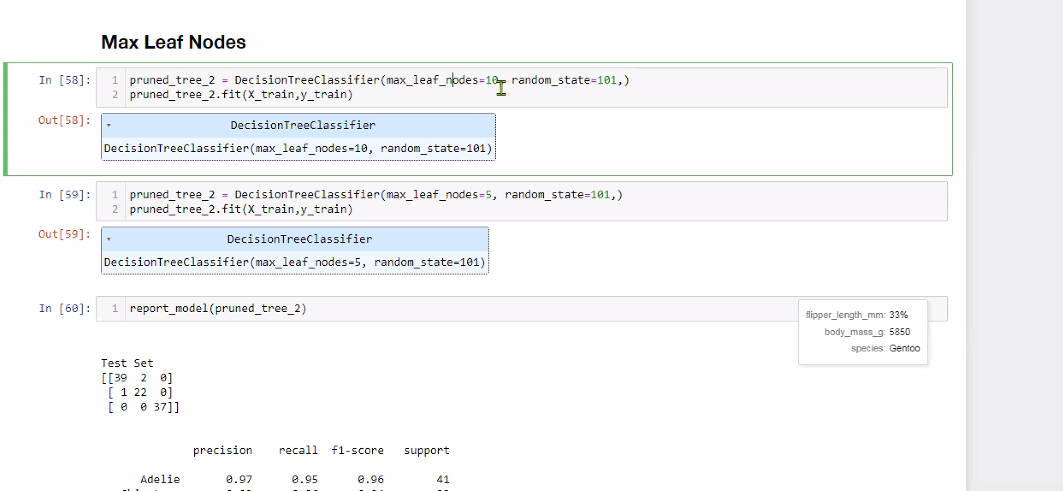


Yeni gelen gozleme ayni sorulari soruyor, ona gore classini belirliyor.

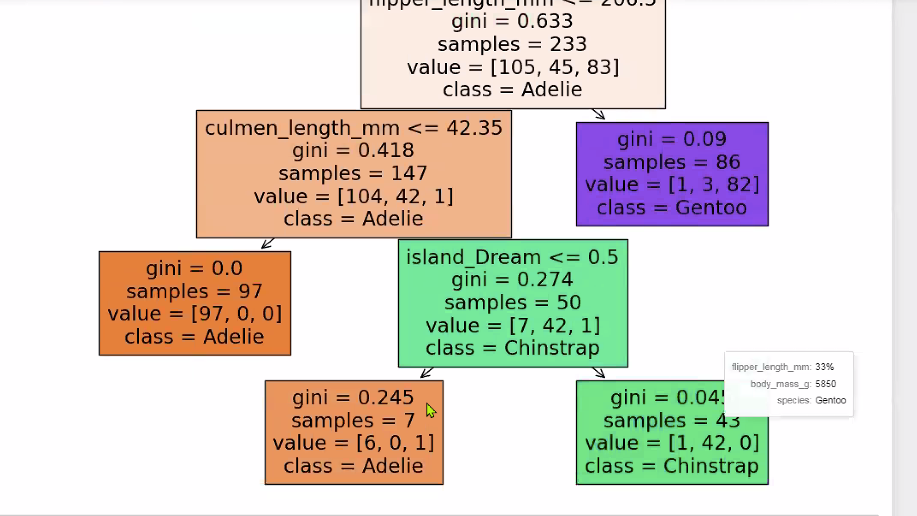
**4 deniyorum:**



5 overfit oldugunu dusunursem. En iyisi 3 diyorum max\_depth icin.



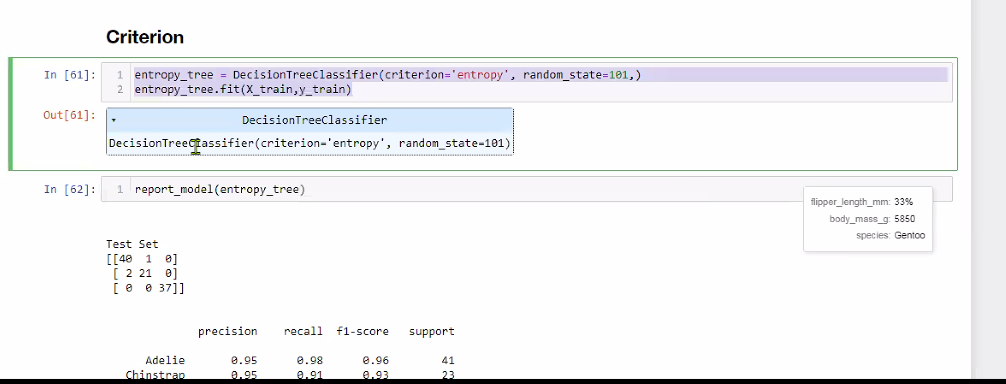
4 tane yaprakla bana sonucu gotur dersem. Max\_leaf\_nodes 4 dedigimde

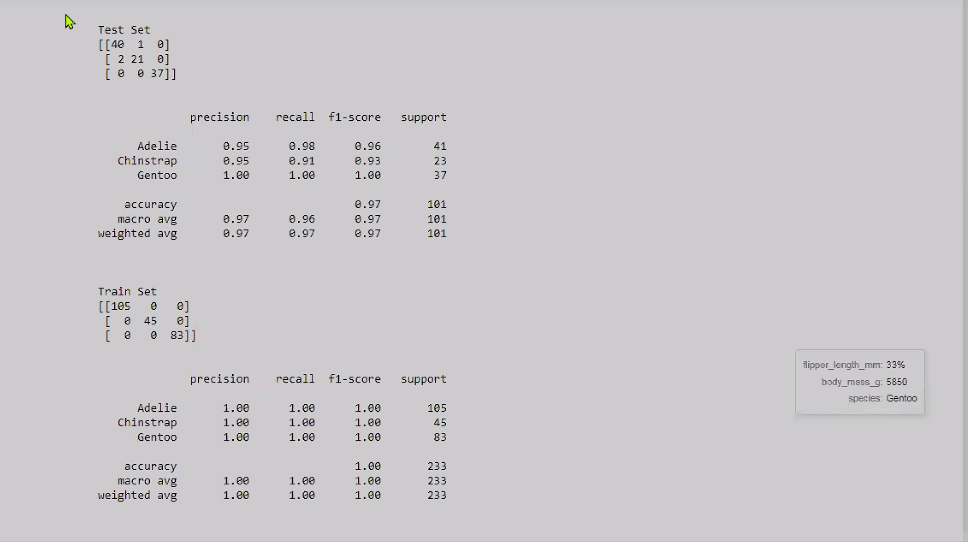


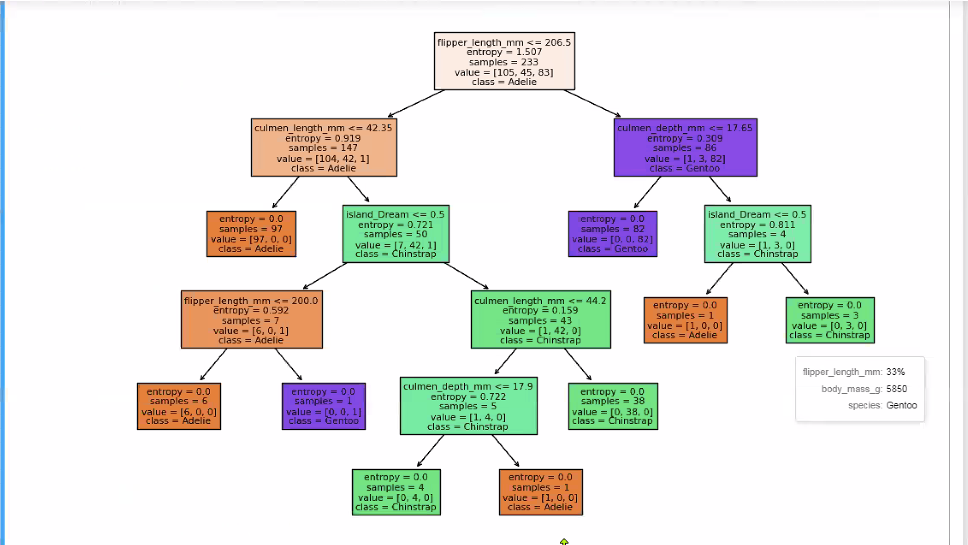
Belli bir yerden sonra hepsini bilecegi icin onemli degil. Mesela 7 de girseniz 15 de girseniz 7 yaprakta bilecegi icin, max\_leaf\_nodes 15 girseydim bile 7 ile bilekceti.

**max\_depth>max\_leaf\_nodes**

Yani ben max\_depth’e 2 verdim, max\_leaf\_nodes’a 5 girdim diyelim. Max\_depth 4 yapraklar sonuca ulastiysa, 4’te keser. Max\_leaf nodes’a 5 girmemin anlami olmaz.

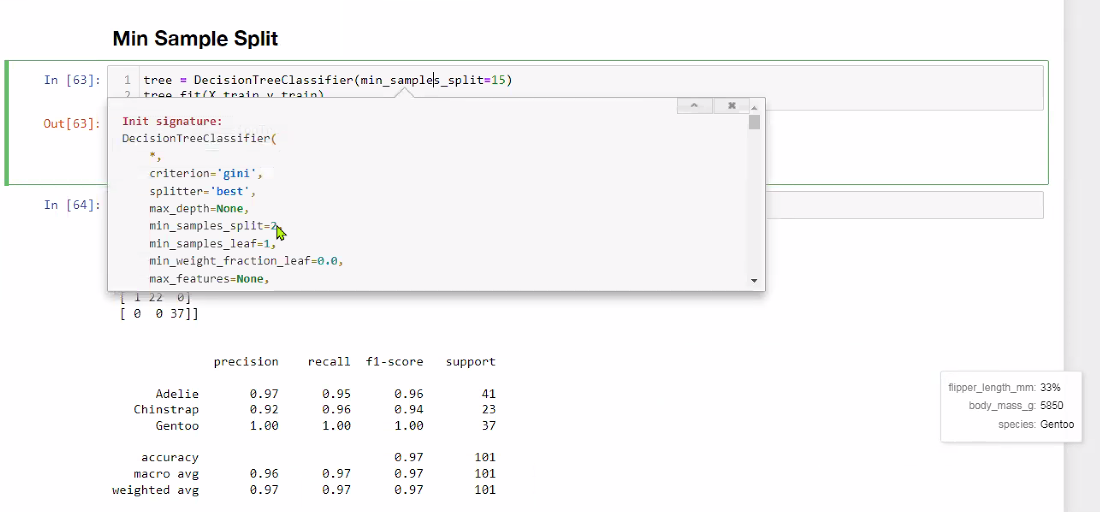


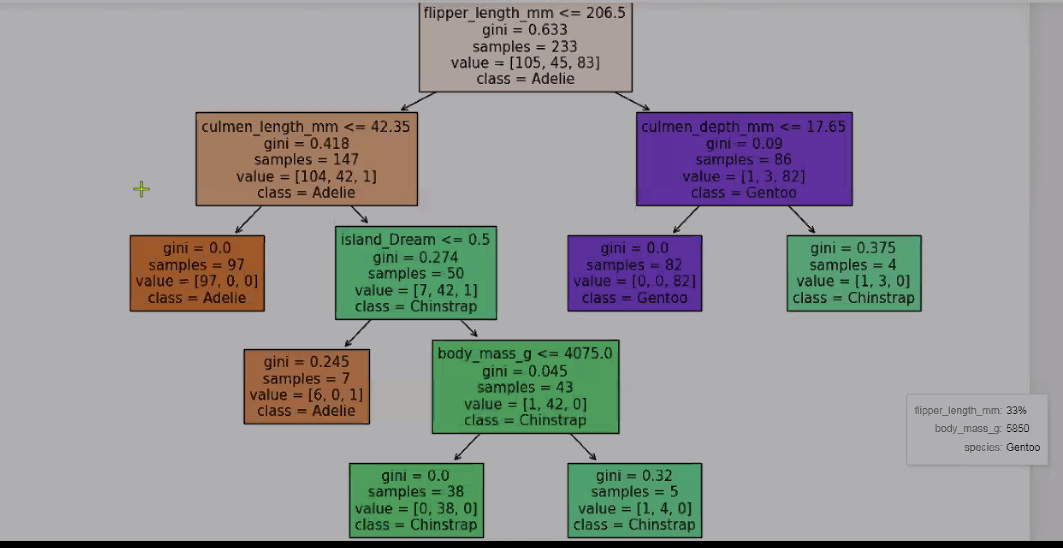




Entrophy tamamen gini ile ayni calisti. Gini entropy’den farkli cikmadi.

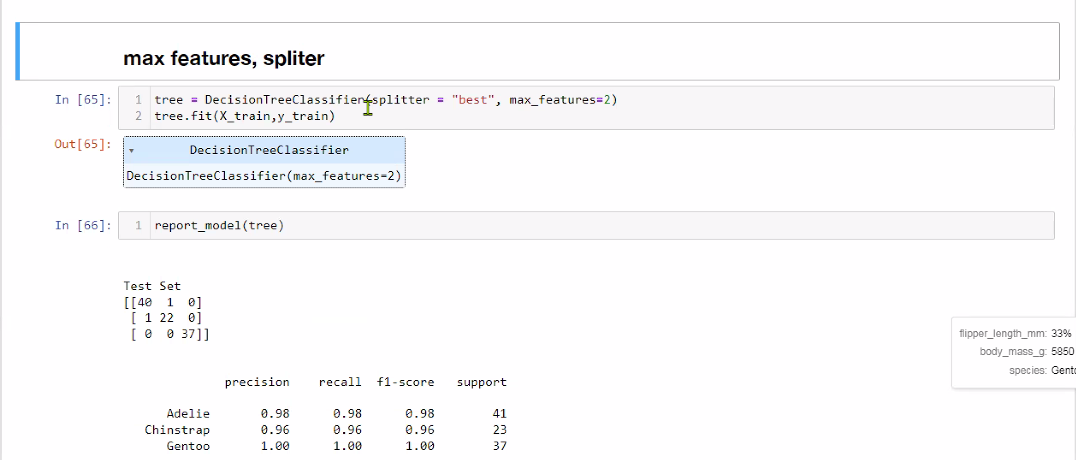
Multiclass olunca entropy’de 1’den buyuk cikabilir.





Min\_sample split--> bir yapragin bolunmeye devam etmesi icin minimum gozlem sayisi

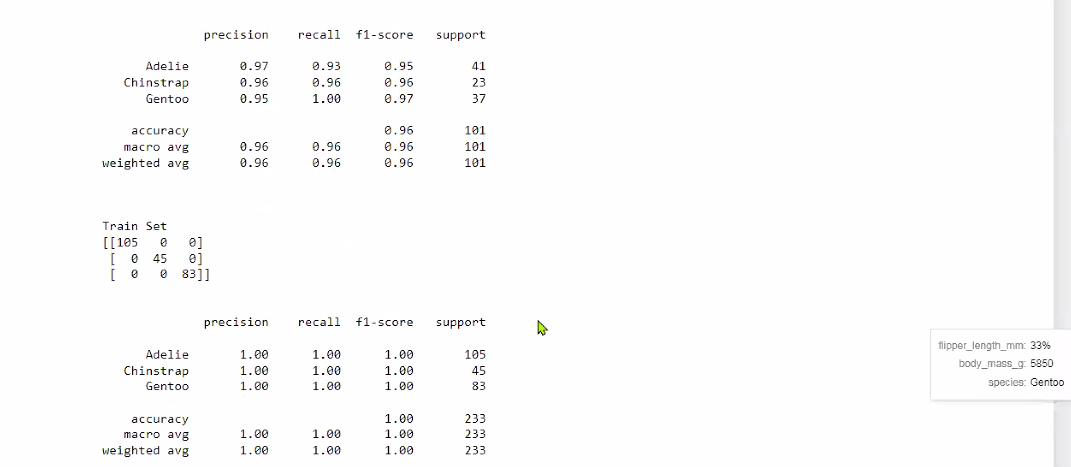
Degeri 5 girdigim icin, 4 gozlem kaldiysa artik bolunmeye devam etmez.



**Splitter=best** dersem en guclu feature’i aliyor ondan basliyor.

Random dersem rastgele bir feature’dan basliyor.

**Best dersem:**



**Max\_features=2** dedigimde random 2 tane secip onlardan best olani ile basliyor.

Herbir decision node’da bu soruyu soruyor. Randomize olarak sectigi bu 2 feature’dan best olani aliyor. Sorularini sorup bir sonraki asamaya gecince de yine random sectigi 2 feature’dan best olani aliyor ona gore

Splitter=best zaten bunun defaultu.

Splitter=random dersem ise, randomly sectigi 2 feature’dan aralarindan da random birini belirler.

