TechCareer - Makine Öğrenimi Bootcamp

Bootcamp süreci boyunca öğrendiğimiz bütün modellerin parametlerinin neler olduğu ve ne işe yaradığı konusuda olan bir araştırma yapacağım bu yazımda.

Makine Öğrenimi Tahmin Modelleri

Linear Regression:

**Parametreler:**

* **fit\_intercept : bool, default = True**

Bu model için kesmenin hesaplanıp hesaplanmayacağı belirlenir. False olarak ayarlanırsa, hesaplamalarda hiçbir engelleme kullanılmaz (yani verilerin ortalanması beklenir).

* **Normalize : bool, default = False**

fit\_intercept, False olarak ayarlandığında bu parametre yok sayılır. True ise, X regresörleri, ortalamanın çıkarılması ve l2-normuna bölünmesiyle regresyondan önce normalize edilecektir.

* **copy\_X : bool, default = True**

True ise, X kopyalanacaktır; yoksa üzerine yazılabilir.

* **n\_jobs : int, default = None**

Hesaplama için kullanılacak iş sayısı. Bu, yalnızca yeterince büyük sorunlar olması durumunda, yani ilk olarak n\_targets > 1 ve ikinci olarak X'in seyrek olması veya pozitifin True olarak ayarlanması durumunda hızlanma sağlayacaktır. Hiçbiri, bir joblib.parallel\_backend bağlamında olmadığı sürece 1 anlamına gelir. -1, tüm işlemcileri kullanmak anlamına gelir.

* **Positive : bool, default=False**

True olarak ayarlandığında, katsayıları pozitif olmaya zorlar. Bu seçenek yalnızca yoğun diziler için desteklenir.

Polynomial Linear Regression

**Parametreler:**

* **Degree : int or tuple (min\_degree, max\_degree), default = 2**

Tek bir int verilirse, polinom özelliklerinin maksimum derecesini belirtir. Bir tanımlama grubu (min\_degree, max\_degree) geçirilirse, min\_degree minimumdur ve max\_degree, oluşturulan özelliklerin maksimum polinom derecesidir. min\_degree=0 ve min\_degree=1 değerlerinin, sıfır derece teriminin çıktısının include\_bias tarafından belirlendiği gibi eşdeğer olduğuna dikkat edin.

* **interaction\_only : bool, default = False**

True ise, yalnızca etkileşim özellikleri üretilir: en fazla derecede farklı girdi özelliklerinin ürünleri olan özellikler, yani aynı girdi özelliğinden 2 veya daha yüksek güce sahip terimler hariç tutulur:

* + Dahil : x[0], x[1], x[0] \* x[1], etc.
  + Hariç tutulan : x[0] \*\* 2, x[0] \*\* 2 \* x[1], etc.
* **include\_bias : bool, default = True**

True ise (varsayılan), tüm polinom güçlerinin sıfır olduğu bir özellik (yani birler sütunu - doğrusal bir modelde bir kesme terimi olarak işlev görür) bir sapma sütunu ekleyin.

* **Order : {‘C’, ‘F’}, default = ’C’**

Yoğun durumda çıktı dizisinin sırası. 'F' sırasının hesaplanması daha hızlıdır, ancak sonraki tahmin edicileri yavaşlatabilir.

Support Vector Machine Regression

**Parametreler:**

* **Kernel : {‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’} or callable, default = ’rbf’**

Algoritmada kullanılacak çekirdek türünü belirtir. Hiçbiri verilmezse, 'rbf' kullanılacaktır. Bir çağrılabilir verilirse, çekirdek matrisini önceden hesaplamak için kullanılır.

* **Degree : int, default = 3**

Polinom çekirdek fonksiyonunun derecesi ('poli'). Diğer tüm çekirdekler tarafından yok sayılır.

* **Gamma : {‘scale’, ‘auto’} or float, default = ’scale’**

'rbf', 'poly' ve 'sigmoid' için çekirdek katsayısı.

* + gamma='scale' (varsayılan) geçirilirse, gama değeri olarak 1 / (n\_features \* X.var()) kullanır,
  + "auto" ise, 1 / n\_features kullanır.
* **Coef0 : float, default = 0.0**

Çekirdek işlevinde bağımsız terim. Yalnızca 'poli' ve 'sigmoid'de anlamlıdır.

* **Tol : float, default = 1e-3**

Durdurma kriteri için tolerans.

* **C : float, default=1.0**

Düzenlileştirme parametresi. Düzenlileştirmenin gücü C ile ters orantılıdır. Kesinlikle pozitif olmalıdır.

* **Epsilon : float, default=0.1**

epsilon-SVR modelinde Epsilon. Gerçek değerden bir mesafe epsilon içinde tahmin edilen noktalarla eğitim kaybı fonksiyonunda hiçbir cezanın ilişkilendirilmediği epsilon tüpünü belirtir.

* **Shrinking : bool, default=True**

Küçülen buluşsal yöntemin kullanılıp kullanılmayacağı.

* **cache\_size : float, default=200**

Çekirdek önbelleğinin boyutunu belirtin (MB cinsinden).

* **Verbose : bool, default=False**

Ayrıntılı çıktıyı etkinleştirin. Bu ayarın, libsvm'de, etkinleştirilirse çok iş parçacıklı bir bağlamda düzgün çalışmayabilecek işlem başına çalışma zamanı ayarından yararlandığını unutmayın.

* **max\_iter : int, default=-1**

Çözücü içindeki yinelemelerde kesin sınır veya sınırsız için -1.

Decision Tree Regressor

**Parametreler:**

* **Criterion : *{“squared\_error”, “friedman\_mse”, “absolute\_error”, “poisson”}, default = ”squared\_error”***

Bir bölünmenin kalitesini ölçme işlevi. Desteklenen kriterler, özellik seçim kriteri olarak varyans azaltımına eşit olan ve potansiyel için Friedman'ın iyileştirme puanı ile ortalama karelenmiş hatayı kullanan “friedman\_mse” her bir uçbirim düğümünün ortalamasını kullanarak L2 kaybını en aza indiren ortalama kare hatası için “kare\_hata”dır. bölmeler, her bir uçbirim düğümünün medyanını kullanarak L1 kaybını en aza indiren ortalama mutlak hata için "mutlak\_hata" ve bölmeleri bulmak için Poisson sapmasındaki azalmayı kullanan "poisson".

* **Splitter : *{“best”, “random”}, default = ”best”***

Her düğümde bölmeyi seçmek için kullanılan strateji. Desteklenen stratejiler, en iyi bölmeyi seçmek için "en iyi" ve en iyi rastgele bölmeyi seçmek için "rastgele"dir.

* **max\_depth : *int, default = None***

Ağacın maksimum derinliği. Hiçbiri ise, tüm yapraklar saf olana veya tüm yapraklar min\_samples\_split örneklerinden daha azını içerene kadar düğümler genişletilir.

* **min\_samples\_split : *int or float, default = 2***

Bir dahili düğümü bölmek için gereken minimum örnek sayısı:

* İnt ise, min\_samples\_split'i minimum sayı olarak kabul edin.
* Float ise, min\_samples\_split bir kesirdir ve ceil(min\_samples\_split \* n\_samples) her bölme için minimum örnek sayısıdır.
* **min\_samples\_leaf : *int or float, default = 1***

Bir yaprak düğümde olması gereken minimum numune sayısı. Herhangi bir derinlikteki bir bölünme noktası, yalnızca sol ve sağ dalların her birinde en az min\_samples\_leaf eğitim örnekleri bırakırsa dikkate alınacaktır. Bu, özellikle regresyonda modeli yumuşatma etkisine sahip olabilir.

* **min\_weight\_fraction\_leaf : *float, default = 0.0***

Bir yaprak düğümde olması gereken ağırlıkların (tüm girdi örneklerinin) toplamının minimum ağırlıklı kesri. Sample\_weight sağlanmadığında numuneler eşit ağırlığa sahiptir.

* **max\_features : *int, float or {“auto”, “sqrt”, “log2”}, default = None***

En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı:

* + İnt ise, her bölmede max\_features özelliklerini göz önünde bulundurun.
  + Float ise, max\_features bir kesirdir ve max(1, int(max\_features \* n\_features\_in\_)) özellikleri her bölmede dikkate alınır.
  + "auto" ise, max\_features=n\_features.
  + "sqrt" ise, max\_features=sqrt(n\_features).
  + "log2" ise, max\_features=log2(n\_features).
  + Hiçbiri ise, max\_features=n\_features.
* **random\_state : *int, RandomState instance or None, default = None***

Tahmincinin rastgeleliğini kontrol eder. Özellikler, ayırıcı "en iyi" olarak ayarlansa bile, her bölmede her zaman rastgele değiştirilir. max\_features < n\_features olduğunda, algoritma aralarındaki en iyi bölünmeyi bulmadan önce her bölmede max\_features'ı rastgele seçecektir. Ancak en iyi bulunan ayrım, max\_features=n\_features olsa bile farklı çalıştırmalarda değişiklik gösterebilir. Bu durumda, kriterin gelişimi birkaç bölme için aynıysa ve rastgele bir bölme seçilmelidir. Uydurma sırasında deterministik bir davranış elde etmek için, random\_state bir tamsayıya sabitlenmelidir.

* **max\_leaf\_nodes : *int, default = None***

max\_leaf\_nodes ile en iyi şekilde bir ağaç büyütün. En iyi düğümler, kirlilikteki göreceli azalma olarak tanımlanır. Hiçbiri ise, sınırsız sayıda yaprak düğümü.

* **min\_impurity\_decrease : *float, default = 0.0***

Bu bölünme, bu değerden daha büyük veya buna eşit safsızlıkta bir azalmaya neden olursa, bir düğüm bölünecektir.

* **ccp\_alpha : *non-negative float, default = 0.0***

Minimum Maliyet-Karmaşıklık Budaması için kullanılan karmaşıklık parametresi. ccp\_alpha'dan daha küçük olan en büyük maliyet karmaşıklığına sahip alt ağaç seçilecektir. Varsayılan olarak, budama yapılmaz.

Random Forest Regressor

**Parametreler:**

* **n\_estimators : int, default = 100**

Ormandaki ağaç sayısı.

* **Criterion : {“squared\_error”, “absolute\_error”, “poisson”}, default = ”squared\_error”**

Bir bölünmenin kalitesini ölçme işlevi. Desteklenen kriterler, özellik seçim kriteri olarak varyans azaltımına eşit olan ortalama karesel hata için "kare\_hata", ortalama mutlak hata için "mutlak\_hata" ve bölmeleri bulmak için Poisson sapmasını kullanan "poisson"dur. "absolute\_error" kullanarak eğitim, "squared\_error" kullanımına göre önemli ölçüde daha yavaştır.

* **max\_depth : int, default = None**

Ağacın maksimum derinliği. Hiçbiri ise, tüm yapraklar saf olana veya tüm yapraklar min\_samples\_split örneklerinden daha azını içerene kadar düğümler genişletilir.

* **min\_samples\_split : int or float, default = 2**

Bir dahili düğümü bölmek için gereken minimum örnek sayısı:

* + İnt ise, min\_samples\_split'i minimum sayı olarak kabul edin.
  + Float ise, min\_samples\_split bir kesirdir ve ceil(min\_samples\_split \* n\_samples) her bölme için minimum örnek sayısıdır.
* **min\_samples\_leaf : int or float, default = 1**

Bir yaprak düğümde olması gereken minimum numune sayısı. Herhangi bir derinlikteki bir bölünme noktası, yalnızca sol ve sağ dalların her birinde en az min\_samples\_leaf eğitim örnekleri bırakırsa dikkate alınacaktır. Bu, özellikle regresyonda modeli yumuşatma etkisine sahip olabilir.

* + int ise, min\_samples\_leaf'i minimum sayı olarak kabul edin.
  + Float ise, min\_samples\_leaf bir kesirdir ve ceil(min\_samples\_leaf \* n\_samples) her düğüm için minimum örnek sayısıdır.
* **min\_weight\_fraction\_leaf : float, default = 0.0**

Bir yaprak düğümde olması gereken ağırlıkların (tüm girdi örneklerinin) toplamının minimum ağırlıklı kesri. Sample\_weight sağlanmadığında numuneler eşit ağırlığa sahiptir.

* **max\_features : {“sqrt”, “log2”, None}, int or float, default = 1.0**

En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı:

İnt ise, her bölmede max\_features özelliklerini göz önünde bulundurun.

Float ise, max\_features bir kesirdir ve max(1, int(max\_features \* n\_features\_in\_)) özellikleri her bölmede dikkate alınır.

"auto" ise, max\_features=n\_features.

"sqrt" ise, max\_features=sqrt(n\_features).

"log2" ise, max\_features=log2(n\_features).

Hiçbiri ise, max\_features=n\_features.

* **max\_leaf\_nodes : int, default = None**

Max\_leaf\_nodes ile ağaçları en iyi şekilde büyütün. En iyi düğümler, kirlilikteki göreceli azalma olarak tanımlanır. Hiçbiri ise, sınırsız sayıda yaprak düğümü.

* **min\_impurity\_decrease : float, default = 0.0**

Bu bölünme, bu değerden daha büyük veya buna eşit safsızlıkta bir azalmaya neden olursa, bir düğüm bölünecektir.

* **Bootstrap : bool, default = True**

Ağaç oluştururken önyükleme örneklerinin kullanılıp kullanılmadığı. False ise, her bir ağacı oluşturmak için tüm veri kümesi kullanılır.

* **oob\_score : bool, default = False**

Genelleme puanını tahmin etmek için torba dışı örneklerin kullanılıp kullanılmayacağı. Yalnızca bootstrap = True ise kullanılabilir.

* **n\_jobs : int, default = None**

Paralel olarak çalıştırılacak iş sayısı. sığdır, tahmin et, karar\_yolu ve uygula hepsi ağaçlar üzerinde paralelleştirilir. Hiçbiri, bir joblib.parallel\_backend bağlamında olmadığı sürece 1 anlamına gelir. -1, tüm işlemcileri kullanmak anlamına gelir. Daha fazla ayrıntı için Sözlük'e bakın.

* **random\_state : int, RandomState instance or None, default = None**

Hem ağaç oluştururken kullanılan örneklerin önyüklemesinin rastgeleliğini (eğer bootstrap=True) hem de her bir düğümde en iyi bölünmeyi ararken dikkate alınacak özelliklerin örneklemesini (eğer max\_features < n\_features ise) kontrol eder.

* **Verbose : int, default = 0**

Uydurma ve tahmin etme sırasında ayrıntı düzeyini kontrol eder.

* **warm\_start : bool, default = False**

True olarak ayarlandığında, topluluğa daha fazla tahmin edici sığdırmak ve eklemek için önceki çağrının çözümünü yeniden kullanın, aksi takdirde tamamen yeni bir ormanı sığdırın.

* **ccp\_alphanon-negative float, default = 0.0**

Minimum Maliyet-Karmaşıklık Budaması için kullanılan karmaşıklık parametresi. ccp\_alpha'dan daha küçük olan en büyük maliyet karmaşıklığına sahip alt ağaç seçilecektir. Varsayılan olarak, budama yapılmaz.

* **max\_samples : int or float, default = None**

Önyükleme True ise, her bir temel tahminciyi eğitmek için X'ten çekilecek örnek sayısı.

Makine Öğrenimi Sınıflandırma Modelleri

Support Vector Machine Classification

**Parametreler:**

* **C : *float, default = 1.0***

Düzenlileştirme parametresi. Düzenlileştirmenin gücü C ile ters orantılıdır. Kesinlikle pozitif olmalıdır.

* **Kernel : *{‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’} or callable, default = ’rbf’***

Algoritmada kullanılacak çekirdek türünü belirtir. Hiçbiri verilmezse, 'rbf' kullanılacaktır. Bir çağrılabilir verilirse, veri matrislerinden çekirdek matrisini önceden hesaplamak için kullanılır; bu matris bir şekil dizisi olmalıdır (n\_samples, n\_samples).

* **Degree : *int, default = 3***

Polinom çekirdek fonksiyonunun derecesi ('poli'). Diğer tüm çekirdekler tarafından yok sayılır.

* **Gamma : *{‘scale’, ‘auto’} or float, default = ’scale’***

'rbf', 'poly' ve 'sigmoid' için çekirdek katsayısı.

* + gamma='scale' (varsayılan) geçirilirse, gama değeri olarak 1 / (n\_features \* X.var()) kullanır,
  + "auto" ise, 1 / n\_features kullanır.
* **Coef0 : *float, default = 0.0***

Çekirdek işlevinde bağımsız terim. Yalnızca 'poli' ve 'sigmoid'de anlamlıdır.

* **Shrinking : *bool, default = True***

Küçülen buluşsal yöntemin kullanılıp kullanılmayacağı.

* **Probability : *bool, default = False***

Olasılık tahminlerinin etkinleştirilip etkinleştirilemeyeceği. Bu, uygun çağrılmadan önce etkinleştirilmelidir, dahili olarak 5 katlı çapraz doğrulama kullandığından bu yöntemi yavaşlatacaktır ve tahmin\_proba tahmin ile tutarsız olabilir.

* **Tol : *float, default = 1e-3***

Durdurma kriteri için tolerans.

* **cache\_size : *float, default = 200***

Çekirdek önbelleğinin boyutunu belirtin (MB cinsinden).

* **class\_weight : *dict or ‘balanced’, default = None***

SVC için i sınıfının C parametresini class\_weight[i]\*C olarak ayarlayın. Verilmezse, tüm sınıfların bir ağırlığı olması gerekir. “Dengeli” mod, girdi verilerindeki sınıf frekanslarıyla ters orantılı ağırlıkları n\_samples / (n\_classes \* np.bincount(y)) olarak otomatik olarak ayarlamak için y değerlerini kullanır.

* **Verbose : *bool, default = False***

Ayrıntılı çıktıyı etkinleştirin. Bu ayarın, libsvm'de, etkinleştirilirse çok iş parçacıklı bir bağlamda düzgün çalışmayabilecek işlem başına çalışma zamanı ayarından yararlandığını unutmayın.

* **max\_iter : *int, default = -1***

Çözücü içindeki yinelemelerde kesin sınır veya sınırsız için -1.

* **decision\_function\_shape : *{‘ovo’, ‘ovr’}, default = ’ovr’***

Diğer tüm sınıflandırıcılar olarak şeklin (n\_samples, n\_classes) bir-vs-rest ('ovr') karar işlevini mi yoksa şekle (n\_samples) sahip olan libsvm'nin orijinal bire-bir ('ovo') karar işlevini mi döndüreceği , n\_classes \* (n\_classes - 1) / 2). Ancak, dahili olarak bire bir ('ovo') modeli eğitmek için her zaman çok sınıflı bir strateji olarak kullanıldığını unutmayın; bir ovr matrisi yalnızca ovo matrisinden oluşturulur. Parametre, ikili sınıflandırma için yoksayılır.

* **break\_ties : *bool, default = False***

true ise, karar\_fonksiyonu='ovr' ve sınıf sayısı > 2 ise, tahmin, karar\_fonksiyonunun güven değerlerine göre bağları koparacaktır; aksi takdirde bağlanan sınıflar arasında birinci sınıf döndürülür. Lütfen, bağları koparmanın basit bir tahmine kıyasla nispeten yüksek bir hesaplama maliyetine sahip olduğunu unutmayın.

* **random\_state : *int, RandomState instance or None, default = None***

Olasılık tahminleri için verileri karıştırmak için sözde rastgele sayı üretimini kontrol eder. Olasılık Yanlış olduğunda yok sayılır. Birden çok işlev çağrısı arasında yeniden üretilebilir çıktı için bir int iletin.

Logistic Regression

**Parametreler:**

* **Penalty : *{‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default = ’l2’***

Cezanın normunu belirtin:

* + 'none': ceza eklenmez;
  + 'l2': bir L2 ceza terimi ekleyin ve bu varsayılan seçimdir;
  + 'l1': bir L1 ceza terimi ekleyin;
  + 'elasticnet': Hem L1 hem de L2 ceza terimleri eklenir.
* **Dual : *bool, default = False***

İkili veya birincil formülasyon. İkili formülasyon sadece liblinear çözücü ile l2 cezası için uygulanmaktadır. n\_samples > n\_features olduğunda dual=False'ı tercih edin.

* **Tol : *float, default = 1e-4***

Durdurma kriterleri için tolerans.

* **C : *float, default = 1.0***

Düzenlileştirme kuvvetinin tersi; pozitif bir şamandıra olmalıdır. Destek vektör makinelerinde olduğu gibi, daha küçük değerler daha güçlü düzenlileştirmeyi belirtir.

* **fit\_intercept : *bool, default = True***

Karar işlevine bir sabitin (a.k.a. sapma veya kesme) eklenmesi gerekip gerekmediğini belirtir.

* **intercept\_scaling : *float, default = 1***

Yalnızca "liblinear" çözücüsü kullanıldığında ve self.fit\_intercept öğesi True olarak ayarlandığında kullanışlıdır. Bu durumda, x [x, self.intercept\_scaling] olur, yani örnek vektörüne intercept\_scaling'e eşit sabit değere sahip bir "sentetik" özellik eklenir. Engelleme, intercept\_scaling \* sentetik\_özellik\_ağırlığı olur.

* **class\_weight : *dict or ‘balanced’, default = None***

{class\_label:weight} biçimindeki sınıflarla ilişkili ağırlıklar. Verilmezse, tüm sınıfların bir ağırlığı olması gerekir.

“Dengeli” mod, girdi verilerindeki sınıf frekanslarıyla ters orantılı ağırlıkları n\_samples / (n\_classes \* np.bincount(y)) olarak otomatik olarak ayarlamak için y değerlerini kullanır.

* **random\_state : *int, RandomState instance, default = None***

Verileri karıştırmak için çözücü == 'sag', 'destan' veya 'liblinear' olduğunda kullanılır.

* **Solver : *{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default = ’lbfgs’***
* Optimizasyon probleminde kullanılacak algoritma. Varsayılan "lbfgs"dir. Bir çözücü seçmek için aşağıdaki hususları göz önünde bulundurmak isteyebilirsiniz:
  + Küçük veri kümeleri için "liblinear" iyi bir seçimdir, oysa "sag" ve "destan" büyük olanlar için daha hızlıdır;
  + Çok sınıflı problemler için yalnızca "newton-cg", "sag", "saga" ve "lbfgs" çok terimli kaybı ele alır;
  + 'liblinear', bire karşı dinlenme şemalarıyla sınırlıdır.
* **max\_iter : *int, default = 100***

Çözücülerin yakınsaması için alınan maksimum yineleme sayısı.

* **multi\_class : *{‘auto’, ‘ovr’, ‘multinomial’}, default = ’auto’***

Seçilen seçenek 'ovr' ise, her etiket için bir ikili problem uygundur. 'Çok terimli' için en aza indirilen kayıp, veriler ikili olduğunda bile tüm olasılık dağılımına uygun çok terimli kayıptır. çözücü='liblinear' olduğunda 'çok terimli' kullanılamaz. 'auto', veriler ikili ise veya çözücü='liblinear' ise 'ovr'yi seçer ve aksi takdirde 'çok terimli'yi seçer.

* **Verbose : *int, default = 0***

liblinear ve lbfgs çözücüler için ayrıntılı, ayrıntı için herhangi bir pozitif sayıya ayarlayın.

* **warm\_start : *bool, default = False***

True olarak ayarlandığında, başlatma olarak sığdırmak için önceki çağrının çözümünü yeniden kullanın, aksi takdirde önceki çözümü silmeniz yeterlidir. liblinear çözücü için işe yaramaz.

* **n\_jobs : *int, default = None***

multi\_class='ovr'” ise, sınıflar üzerinde paralelleştirme yapılırken kullanılan CPU çekirdeği sayısı. Çözücü, 'multi\_class' belirtilmiş olsun veya olmasın, 'liblinear' olarak ayarlandığında bu parametre yok sayılır. Hiçbiri, bir joblib.parallel\_backend bağlamında olmadığı sürece 1 anlamına gelir. -1, tüm işlemcileri kullanmak anlamına gelir.

* **l1\_ratio : *float, default = None***

0 <= l1\_ratio <= 1 ile Elastic-Net karıştırma parametresi. Yalnızca penaltı='elasticnet' olduğunda kullanılır. l1\_ratio=0 ayarı, penaltı='l2' kullanımına eşdeğerdir, l1\_ratio=1 ayarı ise penaltı='l1' kullanımına eşdeğerdir. 0 < l1\_ratio <1 için ceza, L1 ve L2'nin bir kombinasyonudur.

K-Nearest Neighbors(KNN)

**Parametreler:**

* **n\_neighbors : *int, default = 5***

Kneighbors sorguları için varsayılan olarak kullanılacak komşu sayısı.

* **Weights : *{‘uniform’, ‘distance’} or callable, default = ’uniform’***

Tahminde kullanılan ağırlık fonksiyonu. Olası değerler:

* + “uniform”: tek tip ağırlıklar. Her mahalledeki tüm noktalar eşit ağırlıktadır.
  + “distance”: mesafelerinin tersi ile ağırlık noktaları. bu durumda, bir sorgu noktasının daha yakın komşuları, uzaktaki komşulardan daha büyük bir etkiye sahip olacaktır.
  + [callable] : bir dizi mesafeyi kabul eden ve ağırlıkları içeren aynı şekle sahip bir dizi döndüren kullanıcı tanımlı bir işlev.
* **Algorithm : *{‘auto’, ‘ball\_tree’, ‘kd\_tree’, ‘brute’}, default = ’auto’***

En yakın komşuları hesaplamak için kullanılan algoritma:

* + 'ball\_tree' BallTree'yi kullanacak
  + 'kd\_tree' KDTree'yi kullanacak
  + “brute” bir kaba kuvvet araması kullanır.
  + 'auto', fit yöntemine iletilen değerlere göre en uygun algoritmaya karar vermeye çalışacaktır.
* **leaf\_size : *int, default = 30***

BallTree veya KDTree'ye geçirilen yaprak boyutu. Bu, ağaç depolamak için gereken belleğin yanı sıra, inşaat ve sorgulamanın hızını da etkileyebilir. Optimum değer, sorunun doğasına bağlıdır.

* **P : *int, default = 2***

Minkowski metriği için güç parametresi. p = 1 olduğunda, bu manhattan\_mesafesi (l1) ve p = 2 için euclidean\_mesafesi (l2) kullanımına eşdeğerdir. Rastgele p için minkowski\_mesafesi (l\_p) kullanılır.

* **Metric : *str or callable, default = ’minkowski’***

Mesafe hesaplaması için kullanılacak metrik. Varsayılan, p = 2 olduğunda standart Öklid mesafesiyle sonuçlanan "minkowski"dir. Geçerli metrik değerleri için scipy.spatial.distance belgelerine ve Distance\_metrics içinde listelenen metriklere bakın.

Metrik "önceden hesaplanmış" ise, X'in bir uzaklık matrisi olduğu ve sığdırma sırasında kare olması gerektiği varsayılır. X seyrek bir grafik olabilir, bu durumda sadece "sıfır olmayan" elemanlar komşu olarak kabul edilebilir.

Metrik çağrılabilir bir işlevse, girdi olarak 1B vektörleri temsil eden iki dizi alır ve bu vektörler arasındaki mesafeyi gösteren bir değer döndürmesi gerekir. Bu, Scipy'nin metrikleri için işe yarar, ancak metrik adını bir dize olarak geçirmekten daha az verimlidir.

* **metric\_params : *dict, default = None***

Metrik işlevi için ek anahtar sözcük bağımsız değişkenleri.

* **n\_jobs : *int, default = None***

Komşu araması için çalıştırılacak paralel iş sayısı. Hiçbiri, bir joblib.parallel\_backend bağlamında olmadığı sürece 1 anlamına gelir. -1, tüm işlemcileri kullanmak anlamına gelir.

### GridSearch

**Parametreler:**

* **Estimator : *estimator object***

Bunun scikit-learn tahmin edici arayüzünü uyguladığı varsayılır. Tahmin edicinin bir puan işlevi sağlaması veya puanlamanın geçilmesi gerekir.

* **param\_grid : *dict or list of dictionaries***

Anahtar olarak parametre adları (str) olan sözlük ve değerler olarak denenecek parametre ayarları listeleri veya bu tür sözlüklerin bir listesi, bu durumda listedeki her bir sözlüğün kapsadığı ızgaralar araştırılır. Bu, herhangi bir parametre ayarı dizisi üzerinde arama yapılmasını sağlar.

* **Scoring : *str, callable, list, tuple or dict, default = None***

Test setinde çapraz doğrulanmış modelin performansını değerlendirme stratejisi.

Puanlama tek bir puanı temsil ediyorsa, aşağıdakiler kullanılabilir:

* + tek bir dize (bkz. Puanlama parametresi: model değerlendirme kurallarını tanımlama);
  + tek bir değer döndüren bir çağrılabilir (bkz. Puanlama stratejinizi metrik işlevlerden tanımlama).

Puanlama birden fazla puanı temsil ediyorsa, aşağıdakiler kullanılabilir:

* + benzersiz dizelerin bir listesi veya demeti;
  + anahtarların metrik adları ve değerlerin metrik puanlar olduğu bir sözlük döndüren çağrılabilir;
  + anahtar olarak metrik adlara sahip bir sözlük ve a değerleri çağrılabilir.
* **n\_jobs : *int, default = None***

Paralel olarak çalıştırılacak iş sayısı. Hiçbiri, bir joblib.parallel\_backend bağlamında olmadığı sürece 1 anlamına gelir. -1, tüm işlemcileri kullanmak anlamına gelir.

* **Refit : *bool, str, or callable, default = True***

Tüm veri kümesinde bulunan en iyi parametreleri kullanarak bir tahmin ediciyi yeniden takın.

Çoklu metrik değerlendirme için bunun, tahminciyi en sonunda yeniden yerleştirmek için en iyi parametreleri bulmak için kullanılacak puanlayıcıyı belirten bir str olması gerekir.

En iyi tahminciyi seçerken maksimum puan dışında hususlar olduğunda, yeniden yerleştirme, cv\_results\_ tarafından seçilen en iyi\_index\_ değerini döndüren bir işleve ayarlanabilir. Bu durumda, best\_estimator\_ ve best\_params\_, döndürülen best\_index\_ değerine göre ayarlanırken best\_score\_ özelliği kullanılamaz.

Yeniden takılan tahmin edici, best\_estimator\_ özniteliğinde kullanıma sunulur ve doğrudan bu GridSearchCV örneğinde tahminin kullanılmasına izin verir.

Ayrıca çoklu metrik değerlendirme için, best\_index\_, best\_score\_ ve best\_params\_ nitelikleri yalnızca yeniden takma ayarlanmışsa kullanılabilir olacak ve tümü bu özel puanlayıcı ile belirlenecektir.

Çoklu metrik değerlendirme hakkında daha fazla bilgi için puanlama parametresine bakın.

* **Cv : *int, cross-validation generator or an iterable, default = None***

Çapraz doğrulama bölme stratejisini belirler. cv için olası girdiler şunlardır:

* + Yok, varsayılan 5 katlı çapraz doğrulamayı kullanmak için,
  + tamsayı, bir (Stratified)KFold'daki kat sayısını belirtmek için,
  + CV ayırıcı,
  + Yinelenebilir bir verim (tren, test), dizin dizileri olarak bölünür.

Tamsayı/Hiçbiri girdileri için, tahmin edici bir sınıflandırıcıysa ve y ikili veya çok sınıflıysa, StratifiedKFold kullanılır. Diğer tüm durumlarda KFold kullanılır. Bu ayırıcılar shuffle=False ile başlatılır, böylece bölmeler çağrılar arasında aynı olur.

* **Verbose : *int***

Ayrıntıyı kontrol eder: ne kadar yüksekse, o kadar fazla mesaj.

* + >1 : her kat ve parametre adayı için hesaplama süresi görüntülenir;
  + >2 : puan da görüntülenir;
  + >3 : katlama ve aday parametre indeksleri de hesaplamanın başlangıç zamanı ile birlikte görüntülenir.
* **pre\_dispatch : *int, or str, default = ’2\*n\_jobs’***

Paralel yürütme sırasında gönderilen işlerin sayısını kontrol eder. Bu sayıyı azaltmak, CPU'ların işleyebileceğinden daha fazla iş gönderildiğinde bellek tüketiminde patlamayı önlemek için yararlı olabilir. Bu parametre şunlar olabilir:

Hiçbiri, bu durumda tüm işler hemen oluşturulur ve oluşturulur. İşlerin isteğe bağlı olarak oluşturulmasından kaynaklanan gecikmeleri önlemek için bunu hafif ve hızlı çalışan işler için kullanın.

Oluşturulan toplam işlerin tam sayısını veren bir int

'2\*n\_jobs'ta olduğu gibi n\_jobs işlevi olarak bir ifade veren bir str

* **error\_score : *‘raise’ or numeric, default = np.nan***

Tahmin edici uydurmada bir hata oluşursa puana atanacak değer. 'Yükselt' olarak ayarlanırsa, hata ortaya çıkar. Sayısal bir değer verilirse FitFailedWarning yükseltilir. Bu parametre, her zaman hatayı artıracak olan yeniden takma adımını etkilemez.

* **return\_train\_score : *bool, default = False***

False ise, cv\_results\_ niteliği eğitim puanlarını içermez. Eğitim puanlarının hesaplanması, farklı parametre ayarlarının fazla donanım/eksik donanım takasını nasıl etkilediğine dair içgörüler elde etmek için kullanılır. Ancak eğitim setindeki puanların hesaplanması, hesaplama açısından pahalı olabilir ve en iyi genelleme performansını veren parametreleri seçmek için kesinlikle gerekli değildir.

Naive Bayes Classification

GaussianNB

Parametreleri:

* **Priors : *array-like of shape (n\_classes,)***

Sınıfların ön olasılıkları. Belirtilmişse, öncelikler verilere göre ayarlanmaz.

* **var\_smoothing : *float, default = 1e-9***

Hesaplama kararlılığı için varyanslara eklenen tüm özelliklerin en büyük varyansının kısmı.

# BernoulliNB

Parametreleri:

* **Alpha : *float, default = 1.0***

Katkı maddesi (Laplace/Lidstone) yumuşatma parametresi (düzleştirme yok için 0).

* **Binarize : *float or None, default = 0.0***

Örnek özelliklerin ikilileştirilmesi (booleanlarla eşleme) eşiği. Hiçbiri ise, girdinin zaten ikili vektörlerden oluştuğu varsayılır.

* **fit\_prior : *bool, default = True***

Sınıf öncesi olasılıkları öğrenip öğrenmeme. Yanlış ise, ön üniforma kullanılacaktır.

* **class\_prior : *array-like of shape (n\_classes,), default = None***

Sınıfların ön olasılıkları. Belirtilmişse, öncelikler verilere göre ayarlanmaz.

Decision Tree Classification

**Parametreler:**

* **Criterion : *{“gini”, “entropy”, “log\_loss”}, default = ”gini”***

Bir bölünmenin kalitesini ölçme işlevi. Desteklenen kriterler, Gini safsızlığı için "gini" ve Shannon bilgi kazancı için "log\_loss" ve "entropi"dir.

* **Splitter : *{“best”, “random”}, default = ”best”***

Her düğümde bölmeyi seçmek için kullanılan strateji. Desteklenen stratejiler, en iyi bölmeyi seçmek için "en iyi" ve en iyi rastgele bölmeyi seçmek için "rastgele"dir.

* **max\_depth : *int, default = None***

Ağacın maksimum derinliği. Hiçbiri ise, tüm yapraklar saf olana veya tüm yapraklar min\_samples\_split örneklerinden daha azını içerene kadar düğümler genişletilir.

* **min\_samples\_split : *int or float, default = 2***

Bir dahili düğümü bölmek için gereken minimum örnek sayısı:

* + İnt ise, min\_samples\_split'i minimum sayı olarak kabul edin.
  + Float ise, min\_samples\_split bir kesirdir ve ceil(min\_samples\_split \* n\_samples) her bölme için minimum örnek sayısıdır.
* **min\_samples\_leaf : *int or float, default = 1***

Bir yaprak düğümde olması gereken minimum numune sayısı. Herhangi bir derinlikteki bir bölünme noktası, yalnızca sol ve sağ dalların her birinde en az min\_samples\_leaf eğitim örnekleri bırakırsa dikkate alınacaktır. Bu, özellikle regresyonda modeli yumuşatma etkisine sahip olabilir.

int ise, min\_samples\_leaf'i minimum sayı olarak kabul edin.

Float ise, min\_samples\_leaf bir kesirdir ve ceil(min\_samples\_leaf \* n\_samples) her düğüm için minimum örnek sayısıdır.

* **min\_weight\_fraction\_leaf : *float, default = 0.0***

Bir yaprak düğümde olması gereken ağırlıkların (tüm girdi örneklerinin) toplamının minimum ağırlıklı kesri. Sample\_weight sağlanmadığında numuneler eşit ağırlığa sahiptir.

* **max\_features : *int, float or {“auto”, “sqrt”, “log2”}, default = None***

En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı:

* + İnt ise, her bölmede max\_features özelliklerini göz önünde bulundurun.
  + Float ise, max\_features bir kesirdir ve max(1, int(max\_features \* n\_features\_in\_)) özellikleri her bölmede dikkate alınır.
  + "auto" ise, max\_features=sqrt(n\_features).
  + "sqrt" ise, max\_features=sqrt(n\_features).
  + "log2" ise, max\_features=log2(n\_features).
  + Hiçbiri ise, max\_features=n\_features.
* **random\_state : *int, RandomState instance or None, default = None***

Tahmincinin rastgeleliğini kontrol eder. Özellikler, ayırıcı "en iyi" olarak ayarlansa bile, her bölmede her zaman rastgele değiştirilir. max\_features < n\_features olduğunda, algoritma aralarındaki en iyi bölünmeyi bulmadan önce her bölmede max\_features'ı rastgele seçecektir. Ancak en iyi bulunan ayrım, max\_features=n\_features olsa bile farklı çalıştırmalarda değişiklik gösterebilir. Bu durumda, kriterin gelişimi birkaç bölme için aynıysa ve rastgele bir bölme seçilmelidir. Uydurma sırasında deterministik bir davranış elde etmek için, random\_state bir tamsayıya sabitlenmelidir.

* **max\_leaf\_nodes : *int, default = None***

max\_leaf\_nodes ile en iyi şekilde bir ağaç büyütün. En iyi düğümler, kirlilikteki göreceli azalma olarak tanımlanır. Hiçbiri ise, sınırsız sayıda yaprak düğümü.

* **min\_impurity\_decrease : *float, default = 0.0***

Bu bölünme, bu değerden daha büyük veya buna eşit safsızlıkta bir azalmaya neden olursa, bir düğüm bölünecektir.

The weighted impurity decrease equation is the following:

N\_t / N \* (impurity - N\_t\_R / N\_t \* right\_impurity  
 - N\_t\_L / N\_t \* left\_impurity)

burada N toplam örnek sayısıdır, N\_t geçerli düğümdeki örnek sayısıdır, N\_t\_L sol alt öğedeki örnek sayısıdır ve N\_t\_R sağ alt öğedeki örnek sayısıdır.

N, N\_t, N\_t\_R ve N\_t\_L, numune ağırlığı iletildiyse ağırlıklı toplamı ifade eder.

* **class\_weight : *dict, list of dict or “balanced”, default = None***

{class\_label:weight} biçimindeki sınıflarla ilişkili ağırlıklar. Yok ise, tüm sınıfların bir ağırlığı olması gerekir. Çok çıkışlı problemler için, y sütunlarıyla aynı sırada bir dicts listesi sağlanabilir.

Çoklu çıktı (çoklu etiket dahil) için, her sütunun her sınıfı için kendi diktinde ağırlıkların tanımlanması gerektiğini unutmayın. Örneğin, dört sınıflı çok etiketli sınıflandırma için ağırlıklar [{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0:1, 1: olmalıdır. 1}] yerine [{1:1}, {2:5}, {3:1}, {4:1}].

“Dengeli” mod, girdi verilerindeki sınıf frekanslarıyla ters orantılı ağırlıkları n\_samples / (n\_classes \* np.bincount(y)) olarak otomatik olarak ayarlamak için y değerlerini kullanır.

Çoklu çıktı için, her bir y sütununun ağırlıkları çarpılacaktır.

Sample\_weight belirtilirse, bu ağırlıkların sample\_weight ile (uyma yönteminden geçirilen) çarpılacağını unutmayın.

* **ccp\_alpha : *non-negative float, default = 0.0***

Minimum Maliyet-Karmaşıklık Budaması için kullanılan karmaşıklık parametresi. ccp\_alpha'dan daha küçük olan en büyük maliyet karmaşıklığına sahip alt ağaç seçilecektir. Varsayılan olarak, budama yapılmaz. Ayrıntılar için Minimal Maliyet-Karmaşıklık Budama bölümüne bakın.

Random Forest Classification

**Parametreler:**

* **n\_estimators : *int, default = 100***

Ormandaki ağaç sayısı.

* **Criterion : *{“gini”, “entropy”, “log\_loss”}, default = ”gini”***

Bir bölünmenin kalitesini ölçme işlevi. Desteklenen kriterler, Gini safsızlığı için "gini" ve Shannon bilgi kazancı için "log\_loss" ve "entropi"dir, bkz. Matematiksel formülasyon. Not: Bu parametre ağaca özeldir.

* **max\_depth : *int, default = None***

Ağacın maksimum derinliği. Hiçbiri ise, tüm yapraklar saf olana veya tüm yapraklar min\_samples\_split örneklerinden daha azını içerene kadar düğümler genişletilir.

* **min\_samples\_split : *int or float, default = 2***

The minimum number of samples required to split an internal node:

* + İnt ise, min\_samples\_split'i minimum sayı olarak kabul edin.
  + Float ise, min\_samples\_split bir kesirdir ve ceil(min\_samples\_split \* n\_samples) her bölme için minimum örnek sayısıdır.
* **min\_samples\_leaf : *int or float, default = 1***

Bir yaprak düğümde olması gereken minimum numune sayısı. Herhangi bir derinlikteki bir bölünme noktası, yalnızca sol ve sağ dalların her birinde en az min\_samples\_leaf eğitim örnekleri bırakırsa dikkate alınacaktır. Bu, özellikle regresyonda modeli yumuşatma etkisine sahip olabilir.

int ise, min\_samples\_leaf'i minimum sayı olarak kabul edin.

Float ise, min\_samples\_leaf bir kesirdir ve ceil(min\_samples\_leaf \* n\_samples) her düğüm için minimum örnek sayısıdır.

* **min\_weight\_fraction\_leaf : *float, default = 0.0***

Bir yaprak düğümde olması gereken ağırlıkların (tüm girdi örneklerinin) toplamının minimum ağırlıklı kesri. Sample\_weight sağlanmadığında numuneler eşit ağırlığa sahiptir.

* **max\_features : *{“sqrt”, “log2”, None}, int or float, default= ”sqrt”***

En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı:

* + İnt ise, her bölmede max\_features özelliklerini göz önünde bulundurun.
  + Float ise, max\_features bir kesirdir ve max(1, int(max\_features \* n\_features\_in\_)) özellikleri her bölmede dikkate alınır.
  + "auto" ise, max\_features=sqrt(n\_features).
  + "sqrt" ise, max\_features=sqrt(n\_features).
  + "log2" ise, max\_features=log2(n\_features).
  + Hiçbiri ise, max\_features=n\_features.
* **max\_leaf\_nodes : *int, default = None***

Max\_leaf\_nodes ile ağaçları en iyi şekilde büyütün. En iyi düğümler, kirlilikteki göreceli azalma olarak tanımlanır. Hiçbiri ise, sınırsız sayıda yaprak düğümü.

* **min\_impurity\_decrease : *float, default = 0.0***

Bu bölünme, bu değerden daha büyük veya buna eşit safsızlıkta bir azalmaya neden olursa, bir düğüm bölünecektir.

Ağırlıklı safsızlık azalma denklemi aşağıdaki gibidir:

N\_t / N \* (impurity - N\_t\_R / N\_t \* right\_impurity  
 - N\_t\_L / N\_t \* left\_impurity)

burada N toplam örnek sayısıdır, N\_t geçerli düğümdeki örnek sayısıdır, N\_t\_L sol alt öğedeki örnek sayısıdır ve N\_t\_R sağ alt öğedeki örnek sayısıdır.

N, N\_t, N\_t\_R ve N\_t\_L, numune ağırlığı iletildiyse ağırlıklı toplamı ifade eder.

* **Bootstrap : *bool, default = True***

Ağaç oluştururken önyükleme örneklerinin kullanılıp kullanılmadığı. False ise, her bir ağacı oluşturmak için tüm veri kümesi kullanılır.

* **oob\_score : *bool, default = False***

Genelleme puanını tahmin etmek için torba dışı örneklerin kullanılıp kullanılmayacağı. Yalnızca bootstrap=True ise kullanılabilir.

* **n\_jobs : *int, default = None***

Paralel olarak çalıştırılacak iş sayısı. sığdır, tahmin et, karar\_yolu ve uygula hepsi ağaçlar üzerinde paralelleştirilir. Hiçbiri, bir joblib.parallel\_backend bağlamında olmadığı sürece 1 anlamına gelir. -1, tüm işlemcileri kullanmak anlamına gelir. Daha fazla ayrıntı için Sözlük'e bakın.

* **random\_state : *int, RandomState instance or None, default = None***

Hem ağaç oluştururken kullanılan örneklerin önyüklemesinin rastgeleliğini (eğer bootstrap=True) hem de her bir düğümde en iyi bölünmeyi ararken dikkate alınacak özelliklerin örneklemesini (eğer max\_features < n\_features ise) kontrol eder.

* **Verbose : *int, default= 0***

Uydurma ve tahmin etme sırasında ayrıntı düzeyini kontrol eder.

* **warm\_start : *bool, default = False***

True olarak ayarlandığında, topluluğa daha fazla tahmin edici sığdırmak ve eklemek için önceki çağrının çözümünü yeniden kullanın, aksi takdirde tamamen yeni bir ormanı sığdırın.

* **class\_weight : *{“balanced”, “balanced\_subsample”}, dict or list of dicts, default = None***

{class\_label:weight} biçimindeki sınıflarla ilişkili ağırlıklar. Verilmezse, tüm sınıfların bir ağırlığı olması gerekir. Çok çıkışlı problemler için, y sütunlarıyla aynı sırada bir dicts listesi sağlanabilir.

Çoklu çıktı (çoklu etiket dahil) için, her sütunun her sınıfı için kendi diktinde ağırlıkların tanımlanması gerektiğini unutmayın. Örneğin, dört sınıflı çok etiketli sınıflandırma için ağırlıklar [{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0:1, 1: olmalıdır. 1}] yerine [{1:1}, {2:5}, {3:1}, {4:1}].

“balanced” mod, girdi verilerindeki sınıf frekanslarıyla ters orantılı ağırlıkları n\_samples / (n\_classes \* np.bincount(y)) olarak otomatik olarak ayarlamak için y değerlerini kullanır.

" balanced\_subsample" modu, ağırlıkların büyütülen her ağaç için önyükleme örneğine göre hesaplanması dışında "dengeli" ile aynıdır.

Çoklu çıktı için, her bir y sütununun ağırlıkları çarpılacaktır.

Sample\_weight belirtilirse, bu ağırlıkların sample\_weight ile (uyma yönteminden geçirilen) çarpılacağını unutmayın.

* **ccp\_alpha : *non-negative float, default = 0.0***

Minimum Maliyet-Karmaşıklık Budaması için kullanılan karmaşıklık parametresi. ccp\_alpha'dan daha küçük olan en büyük maliyet karmaşıklığına sahip alt ağaç seçilecektir. Varsayılan olarak, budama yapılmaz. Ayrıntılar için Minimal Maliyet-Karmaşıklık Budama bölümüne bakın.

* **max\_samples : *int or float, default = None***

Önyükleme True ise, her bir temel tahminciyi eğitmek için X'ten çekilecek örnek sayısı.

* + Hiçbiri (varsayılan) ise, X.shape[0] örnekleri çizin.
  + İnt ise, max\_samples örnekleri çizin.
  + Yüzerse, max\_samples \* X.shape[0] örnekleri çizin. Bu nedenle, max\_samples (0.0, 1.0] aralığında olmalıdır.

Makine Öğrenmesi Veri Seti Küçültme Yöntemleri

Principal component analysis (PCA)

**Parametleri:**

* **n\_components : *int, float or ‘mle’, default = None***

Tutulacak bileşenlerin sayısı. n\_components ayarlanmazsa tüm bileşenler korunur:

n\_components == min(n\_samples, n\_features)

n\_components == 'mle' ve svd\_solver == 'dolu' ise, boyutu tahmin etmek için Minka'nın MLE'si kullanılır. n\_components == 'mle' kullanımı, svd\_solver == 'auto' ifadesini svd\_solver == 'full' olarak yorumlayacaktır.

0 < n\_components < 1 ve svd\_solver == 'dolu' ise, açıklanması gereken varyans miktarı n\_components tarafından belirtilen yüzdeden daha büyük olacak şekilde bileşen sayısını seçin.

svd\_solver == 'arpack' ise, bileşenlerin sayısı, n\_features ve n\_samples minimumundan kesinlikle az olmalıdır.

Hence, the None case results in:

n\_components == min(n\_samples, n\_features) - 1

* **Copy : *bool, default = True***

False ise, sığdırmak için iletilen verilerin üzerine yazılır ve fit(X).transform(X)'i çalıştırmak beklenen sonuçları vermez, bunun yerine fit\_transform(X) kullanın.

* **Whiten : *bool, default = False***

True (varsayılan olarak False) olduğunda, component\_ vektörleri n\_samples'ın kareköküyle çarpılır ve ardından birim bileşen bazında varyanslarla ilişkisiz çıktılar sağlamak için tekil değerlere bölünür.

Beyazlatma, dönüştürülmüş sinyalden (bileşenlerin göreli varyans ölçekleri) bazı bilgileri kaldıracaktır, ancak bazen, verilerinin bazı kablolu varsayımlara uymasını sağlayarak aşağı akış tahmincilerinin tahmin doğruluğunu iyileştirebilir.

* **svd\_solver : *{‘auto’, ‘full’, ‘arpack’, ‘randomized’}, default = ’auto’***
  + **If auto :**

Çözücü, X.shape ve n\_components'a dayalı bir varsayılan ilke tarafından seçilir: giriş verileri 500x500'den büyükse ve çıkarılacak bileşen sayısı verilerin en küçük boyutunun %80'inden azsa, o zaman daha verimli "rastgele" ' yöntemi etkinleştirildi. Aksi takdirde, tam SVD hesaplanır ve daha sonra isteğe bağlı olarak kesilir.

* + **If full :**

scipy.linalg.svd aracılığıyla standart LAPACK çözücüyü çağırarak tam SVD'yi çalıştırın ve son işleme ile bileşenleri seçin

* + **If arpack :**

scipy.sparse.linalg.svds aracılığıyla ARPACK çözücüyü çağıran n\_components'a kesilmiş SVD'yi çalıştırın. Kesinlikle 0 < n\_components < min(X.shape) gerektirir

* + **If randomized :**

Halko ve ark.'nın yöntemiyle randomize SVD çalıştırın.

* **Tol : *float, default = 0.0***

svd\_solver == 'arpack' tarafından hesaplanan tekil değerler için tolerans. [0.0, sonsuz) aralığında olmalıdır.

* **iterated\_power : *int or ‘auto’, default = ’auto’***

svd\_solver == “randomized” tarafından hesaplanan güç yöntemi için yineleme sayısı. [0, sonsuz) aralığında olmalıdır.

* **n\_oversamples : *int, default = 10***

Bu parametre yalnızca svd\_solver="randomized" olduğunda geçerlidir. Uygun koşullandırmayı sağlamak için X aralığını örneklemek için ek rastgele vektör sayısına karşılık gelir. Daha fazla ayrıntı için randomize\_svd'ye bakın.

* **power\_iteration\_normalizer : *{‘auto’, ‘QR’, ‘LU’, ‘none’}, default = ’auto’***

Rastgele SVD çözücü için güç yineleme normalleştiricisi. ARPACK tarafından kullanılmaz. Daha fazla ayrıntı için randomize\_svd'ye bakın.

* **random\_state : *int, RandomState instance or None, default = None***

**'arpack' veya ' randomized' çözücüler kullanıldığında kullanılır. Birden çok işlev çağrısında tekrarlanabilir sonuçlar için bir int iletin.**

Linear Discriminant Analysis

**Parametreler:**

* **Solver : *{‘svd’, ‘lsqr’, ‘eigen’}, default = ’svd’***

**Solver to use, possible values:**

* + 'svd': Tekil değer ayrıştırması (varsayılan). Kovaryans matrisini hesaplamaz, bu nedenle bu çözücü, çok sayıda özelliği olan veriler için önerilir.
  + 'lsqr': En küçük kareler çözümü. Büzülme veya özel kovaryans tahmincisi ile birleştirilebilir.
  + 'eigen': Özdeğer ayrıştırması. Büzülme veya özel kovaryans tahmincisi ile birleştirilebilir.
* **Shrinkage : *‘auto’ or float, default = None***

**Shrinkage parameter, possible values:**

* + Yok: büzülme yok (varsayılan).
  + 'auto': Ledoit-Wolf lemmasını kullanarak otomatik küçültme.
  + float between 0 and 1: fixed shrinkage parameter.
* **Priors : *array-like of shape (n\_classes,), default = None***

Sınıf ön olasılıkları. Varsayılan olarak, sınıf oranları eğitim verilerinden çıkarılır.

* **n\_components : *int, default = None***

Boyut azaltma için bileşen sayısı (<= min(n\_classes - 1, n\_features)). Yok ise, min(n\_classes - 1, n\_features) olarak ayarlanacaktır. Bu parametre yalnızca dönüştürme yöntemini etkiler.

* **store\_covariance : *bool, default = False***

True ise, çözücü 'svd' olduğunda ağırlıklı sınıf içi kovaryans matrisini açıkça hesaplayın. Matris her zaman diğer çözücüler için hesaplanır ve saklanır.

* **Tol : *float, default = 1.0e-4***

X'in tekil değerinin anlamlı kabul edilmesi için mutlak eşik, X'in sıralamasını tahmin etmek için kullanılır. Tekil değerleri önemsiz olan boyutlar atılır. Yalnızca çözücü "svd" ise kullanılır.

* **covariance\_estimator : *covariance estimator, default = None***

Yok değilse, kovaryans matrislerini tahmin etmek için deneysel kovaryans tahmin edicisine (potansiyel küçülme ile) güvenmek yerine kovaryans\_tahmincisi kullanılır. Nesnenin, sklearn.covariance'daki tahmin ediciler gibi bir uygun yöntemi ve bir kovaryans\_ özniteliği olmalıdır. Yok ise, büzülme parametresi tahmini yönlendirir.

Büzülme kullanılıyorsa bu, Yok olarak bırakılmalıdır. Kovaryance\_estimator'ın yalnızca "lsqr" ve "eigen" çözücülerle çalıştığını unutmayın.

Makine Öğrenimi Sınıflandırma Modelleri

K-Means

**Parametreleri:**

* **n\_clusters : *int, default = 8***

Oluşturulacak merkez sayısı kadar oluşturulacak küme sayısı.

* **Init : *{‘k-means++’, ‘random’}, callable or array-like of shape (n\_clusters, n\_features), default = ’k-means++’***

Başlatma yöntemi:

'k-means++' : noktaların genel eylemsizliğe katkısının ampirik olasılık dağılımına dayalı olarak örneklemeyi kullanarak ilk küme merkezlerini seçer. Bu teknik yakınsamayı hızlandırır ve teorik olarak O(log k) -optimal olduğu kanıtlanmıştır. Daha fazla ayrıntı için n\_init açıklamasına bakın.

'rastgele': ilk ağırlık merkezleri için verilerden rastgele n\_clusters gözlemleri (satırlar) seçin.

Bir dizi geçirilirse, şeklinde olmalıdır (n\_clusters, n\_features) ve ilk merkezleri verir.

Bir çağrılabilir iletilirse, X, n\_clusters argümanlarını ve rastgele bir durumu almalı ve bir başlatma döndürmelidir.

* **n\_init : *int, default = 10***

k-ortalamalar algoritmasının farklı centroid tohumlarıyla çalıştırılacağı sürenin sayısı. Nihai sonuçlar, eylemsizlik açısından n\_init ardışık çalıştırmalarının en iyi çıktısı olacaktır.

* **max\_iter : *int, default = 300***

Tek bir çalıştırma için k-ortalama algoritmasının maksimum yineleme sayısı.

* **Tol : *float, default = 1e-4***

Yakınsama beyan etmek için ardışık iki yinelemenin küme merkezlerindeki farkın Frobenius normu ile ilgili göreli tolerans.

* **Verbose : *int, default = 0***

Ayrıntı modu.

* **random\_state : *int, RandomState instance or None, default = None***

Merkezi başlatma için rasgele sayı üretimini belirler. Rastgeleliği deterministik yapmak için bir int kullanın.

* **copy\_x : *bool, default = True***

Mesafeleri önceden hesaplarken, önce verileri ortalamak sayısal olarak daha doğrudur. copy\_x True ise (varsayılan), orijinal veriler değiştirilmez. False ise, orijinal veri değiştirilir ve işlev geri dönmeden önce geri konur, ancak veri ortalamasının çıkarılması ve ardından eklenmesiyle küçük sayısal farklılıklar ortaya çıkabilir. Orijinal veri C-bitişik değilse, copy\_x False olsa bile bir kopyanın yapılacağını unutmayın. Orijinal veriler seyrekse ancak CSR biçiminde değilse, copy\_x False olsa bile bir kopya oluşturulur.

* **Algorithm : *{“lloyd”, “elkan”, “auto”, “full”}, default = ”lloyd”***

K-kullanılacak algoritma anlamına gelir. Klasik EM tarzı algoritma "lloyd" dur. "Elkan" varyasyonu, üçgen eşitsizliği kullanılarak, iyi tanımlanmış kümelere sahip bazı veri kümelerinde daha verimli olabilir. Bununla birlikte, fazladan bir şekil dizisinin (n\_samples, n\_clusters) tahsis edilmesi nedeniyle daha fazla bellek yoğundur.

Agglomerative

**Parametreler:**

* **n\_clusters : *int or None, default = 2***

Bulunacak küme sayısı. Mesafe\_eşiği Yok değilse Yok olmalıdır.

* **Affinity : *str or callable, default = ’euclidean’***

Bağlantıyı hesaplamak için kullanılan metrik. "Öklid", "l1", "l2", "manhattan", "kosinüs" veya "önceden hesaplanmış" olabilir. Bağlantı "koğuş" ise, yalnızca "öklid" kabul edilir. "Önceden hesaplanmışsa", uyum yöntemi için girdi olarak bir uzaklık matrisi (benzerlik matrisi yerine) gereklidir.

* **Memory : *str or object with the joblib.Memory interface, default = None***

Ağacın hesaplanmasının çıktısını önbelleğe almak için kullanılır. Varsayılan olarak, önbelleğe alma yapılmaz. Bir dize verilirse, önbelleğe alma dizinine giden yoldur.

* **Connectivity : *array-like or callable, default = None***

Bağlantı matrisi. Verilerin belirli bir yapısını takip eden komşu örnekleri her örnek için tanımlar. Bu, bir bağlantı matrisinin kendisi veya verileri kneighbors\_graph'tan türetilen gibi bir bağlantı matrisine dönüştüren bir çağrılabilir olabilir. Varsayılan Yok'tur, yani hiyerarşik kümeleme algoritması yapılandırılmamıştır.

* **compute\_full\_tree : *‘auto’ or bool, default = ’auto’***

n\_clusters'da ağaç yapımını erken durdurun. Bu, küme sayısı örnek sayısına kıyasla küçük değilse, hesaplama süresini azaltmak için yararlıdır. Bu seçenek yalnızca bir bağlantı matrisi belirtilirken kullanışlıdır. Ayrıca, küme sayısını değiştirirken ve önbelleğe alma kullanırken, tam ağacı hesaplamanın avantajlı olabileceğini unutmayın. Mesafe\_eşiği Yok değilse Doğru olmalıdır. Varsayılan olarak hesaplama\_tam\_ağacı "otomatik"tir; bu, Distance\_threshold Yok olmadığında veya n\_clusters 100 veya 0,02 \* n\_samples arasındaki maksimum değerden düşük olduğunda True'ya eşdeğerdir. Aksi takdirde, “auto” False ile eşdeğerdir.

* **Linkage : *{‘ward’, ‘complete’, ‘average’, ‘single’}, default = ’ward’***

Hangi bağlantı kriterinin kullanılacağı. Bağlantı kriteri, gözlem kümeleri arasında hangi mesafenin kullanılacağını belirler. Algoritma, bu kriteri en aza indiren küme çiftlerini birleştirecektir.

* + 'ward', birleştirilen kümelerin varyansını en aza indirir.
  + 'ortalama', iki kümenin her bir gözleminin mesafelerinin ortalamasını kullanır.
  + 'tam' veya 'maksimum' bağlantı, iki kümenin tüm gözlemleri arasındaki maksimum mesafeleri kullanır.
  + 'tek', iki kümenin tüm gözlemleri arasındaki minimum mesafeyi kullanır.
* **distance\_threshold : *float, default = None***

Üzerinde kümelerin birleştirilemeyeceği bağlantı mesafesi eşiği. Hiçbiri değilse, n\_clusters Yok ve hesaplama\_full\_ağacı True olmalıdır.

* **compute\_distances : *bool, default = False***

Distance\_threshold kullanılmasa bile kümeler arasındaki mesafeleri hesaplar. Bu, dendrogram görselleştirmesi yapmak için kullanılabilir, ancak bir hesaplama ve bellek yükü sunar.