**ML SESSION -4**

**Regularization:**

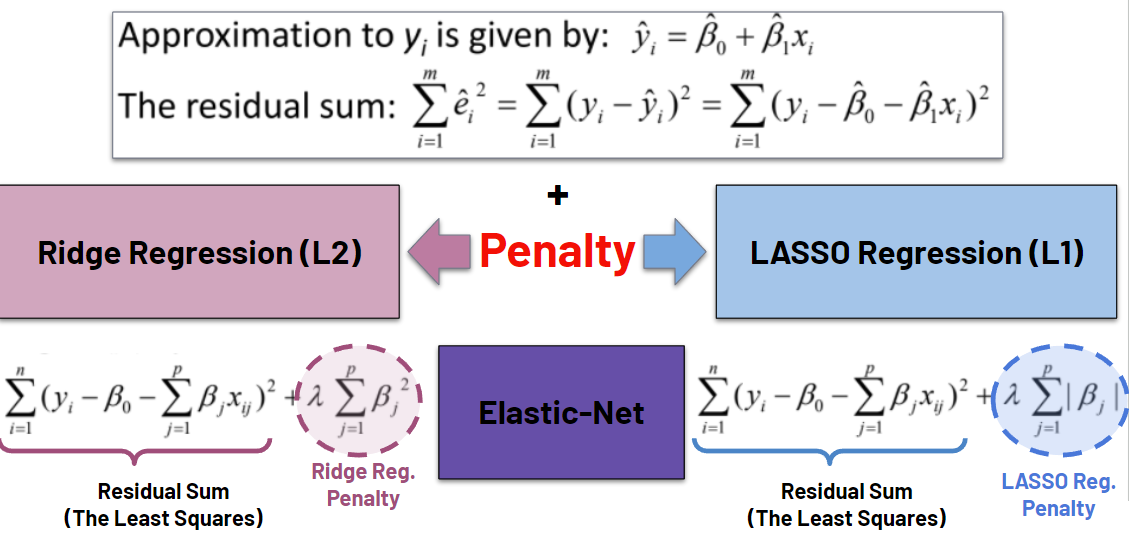
Underfittingle baş etmek zaten kolay. Feature sayısını azaltırsın. Fakat overfitting durumunda ne yapman gerekir? ML'de Regularization, modelin aşırı uymasını (overfitting) önlemek için kullanılan bir yöntemdir. Bu yöntem, modelin karmaşıklığını azaltarak genelleştirme performansını artırmayı hedefler.

Regularization, genellikle bir düzenleme terimi (regularization term) ekleyerek gerçekleştirilir. Bu terim, modelin karmaşıklığına bağlı olarak artar ve modelin kayıplarını artırarak aşırı uyumu önler.

En yaygın kullanılan regularization yöntemleri şunlardır:

1. **L1 Regularization (Lasso):** Bu yöntemde, **regularization term, modeldeki katsayıların mutlak değerlerinin toplamıdır**. Bu yöntem, bazı özelliklerin etkisiz hale getirilmesine neden olur ve böylece seçici özellik seçimine (feature selection) katkıda bulunur.
2. **L2 Regularization (Ridge):** Bu yöntemde, **regularization term, modeldeki katsayıların karelerinin toplamıdır.** Bu yöntem, aşırı uyumun önlenmesinde etkilidir ve tüm özelliklerin önemini korur.
3. **Elastic Net Regularization**: Bu yöntem, L1 ve L2 düzenleme terimlerinin bir kombinasyonunu kullanır. Bu yöntem, Lasso'nun seçici özellik seçimine katkıda bulunmasını ve Ridge'nin tüm özelliklerin önemini korumasını birleştirir.
4. **Dropout**: Bu yöntem, sinir ağları gibi derin öğrenme modellerinde kullanılır. Rastgele olarak bir bölümünü devre dışı bırakarak, modelin aşırı uyumu önlenir ve genelleştirme performansı artırılır.
5. **Early Stopping**: Bu yöntem, eğitim sürecini erken durdurarak aşırı uyumu önlemeyi amaçlar. Eğitim süreci, belirli bir hata eşiği aşıldığında veya doğruluk oranı belirli bir seviyeye ulaştığında sonlandırılır. Bu yöntem, özellikle derin öğrenme modellerinde yaygın olarak kullanılır.

Lineer regresyonda bağımsız değişkenler arasında bir korelasyon varsa, çoklu doğrusallık meydana gelir. Bu problemle karşılaşıldığında, tahminler dengesiz olabilir, model parametreleri için beklenen işaret ve tahminlerden farklı olabilir, parametre varyansları tahmin edilenden daha büyük olabilir ve verilerdeki çok küçük değişiklikler bile parametrelerde büyük farklılıklar gösterebilir.

****

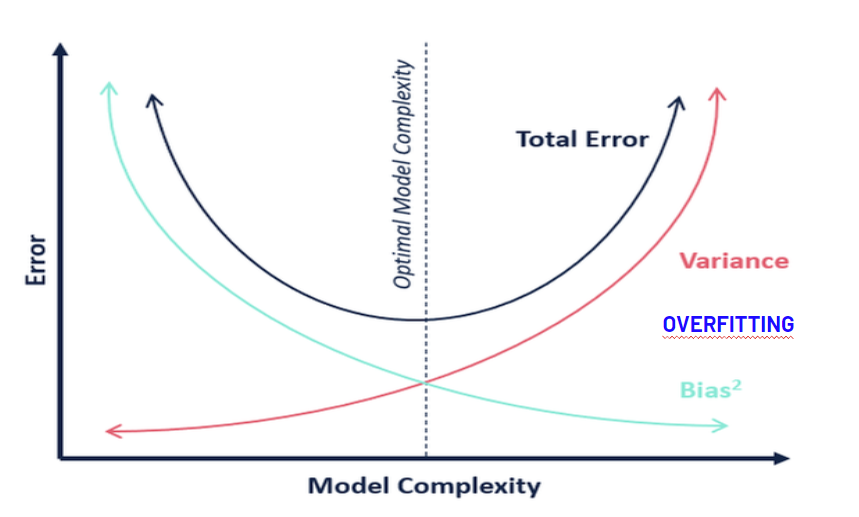
**Overfitting – Underfitting durumlarında lambdanın etkisi:**

**Overfitting** durumunda lambda ile modele hata ekliyoruz. Bu hatayı arttırdıkça (lambdayı büyüttükçe) variance azlıyor ve bias artıyor, yani overfitting azalıyor. Belli bir seviyede bias-varias trade off oluşuyor (optimum seviye yakalanıyor).

Bu noktadan sonra hata eklemeye devam ederseniz yani model compexcity’sini azaltırsanız, variance azalmaya ve bias artmaya devam eder ve bu underfitting’e doğru kaymayan neden olur.

Tersten düşünürsek; underfitting durumunda (yüksek hatada) modelin hatasını azalta azalta giderken (modelin complexcity'sini arttırırken) bir noktada optimum seviyeyi yakalıyoruz (trade off).

Bu noktadan sonra hatayı azaltmaya devam edersek (mesela feature sayısını arttırırsak) bu sefer **overfitting**'e kayıyor (model aşırı öğreniyor)



**parametre** = modelin kendi bulduğu (coef)

**hyperparametre** = benim ayarladığım (lambda)

total errr = bias + variance + ie

ie değişmez. total err değişmez. Dolayısıyla bias artarsa variance azalır. Tam terside geçerli.

**Lamda yı büyütmek regülarizasyonu artırmak ile aynı manaya geliyor.**

**Ridge regression:**

Ridge regresyonu, çoklu doğrusal regresyon problemlerinde kullanılan bir regresyon yöntemidir. Bu yöntem, veri setindeki bağımsız değişkenler arasındaki çoklu doğrusal bağlantı sorunlarını ele alır ve aşırı öğrenmeyi azaltmak için bir düzenleme terimi ekler.

**Ridge regresyonunun özellikleri şunlardır:**

1. Düzenleme Terimi(regularization term): Ridge regresyonu, modelin karmaşıklığına bağlı olarak bir düzenleme terimi ekler (**lambda**). Bu terim, modelin katsayılarının büyüklüğünü sınırlayarak aşırı öğrenmeyi azaltmaya yardımcı olur.
2. Multi Lineer (çoklu doğrusal) Bağlantı Sorununu Azaltma: Ridge regresyonu, veri setindeki bağımsız değişkenler arasındaki multi lineer bağlantı sorunlarını ele alır. Bu sorun, bir veya daha fazla bağımsız değişkenin diğer bağımsız değişkenlerle yüksek bir korelasyona sahip olduğu durumlarda ortaya çıkar. Bu durumda, ridge regresyonu katsayıları azaltarak bu sorunu azaltır.
3. Katsayıların Sıfıra Yaklaşması: Ridge regresyonu, lambdayı arttırdıkça katsayıları **sıfıra yaklaştırır**. Bu özellik, modeldeki **gereksiz bağımsız değişkenleri ortadan kaldırarak** modelin genel performansını arttırmaya yardımcı olur.
4. Veri Ölçeklendirme: Ridge regresyonu, verilerin ölçeğine duyarlıdır. Bu nedenle, verilerin ölçeğini birbirine yakın hale getirmek veya normalleştirmek, modelin performansını arttırmaya yardımcı olabilir.
5. Lambda Değerinin Seçimi: Ridge regresyonunda, lambda adı verilen düzenleme terimi için uygun bir değer seçmek önemlidir. Lambda değeri, aşırı öğrenme ve modele uygunluk arasındaki dengeyi belirler. İdeal lambda değeri, çapraz doğrulama yöntemleri kullanılarak belirlenebilir.

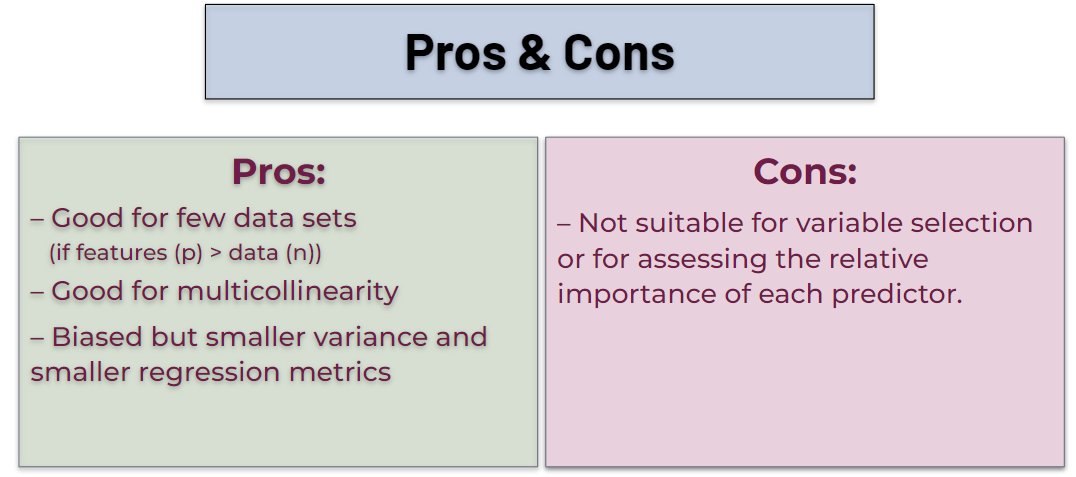
Ridge muhakkak her feature’a bir katsayı veriyor.

**Ridge**, bize **featurelar için katsayıları belirliyor**. Ama muhakkak bütün featurelar için bunu yapıyor. Bir feature selection yapmıyor.

**Lassa** ise **bazı katsayıları sıfırlayıp feature selection yapıyor**. Dolayısıyla diğer featurelar ile yoluna devam ediyor. Lassa cezayı ağır kesiyor.

Not: Notebook (Python) kısmında Lambda’ya biz alpha diyeceğiz. Ancak “lambda” python’da bir fonksiyon türü olduğundan kavramların birbirine karışmaması için scikit-learn’de lambda yerine “alpha” terimi kullanılıyor.

**Ridge regresyonun avantaj/dezavantajları:**



**LASSO regression:**

LASSO (**Least Absolute Shrinkage and Selection Operator**), çoklu doğrusal regresyon problemlerinde kullanılan bir regresyon yöntemidir. LASSO, Ridge regresyonuna benzer bir düzenleme terimi kullanarak aşırı öğrenmeyi azaltır ve aynı zamanda veri setindeki gereksiz bağımsız değişkenleri de elemeye yardımcı olur.

LASSO, lambda regularization term/penalty term (düzenleme terimi/lambda)  kullanarak **katsayıları sıfıra yaklaştırır**. Bu özellik, modeldeki gereksiz bağımsız değişkenleri **ortadan kaldırarak**modelin genel performansını arttırmaya yardımcı olur.

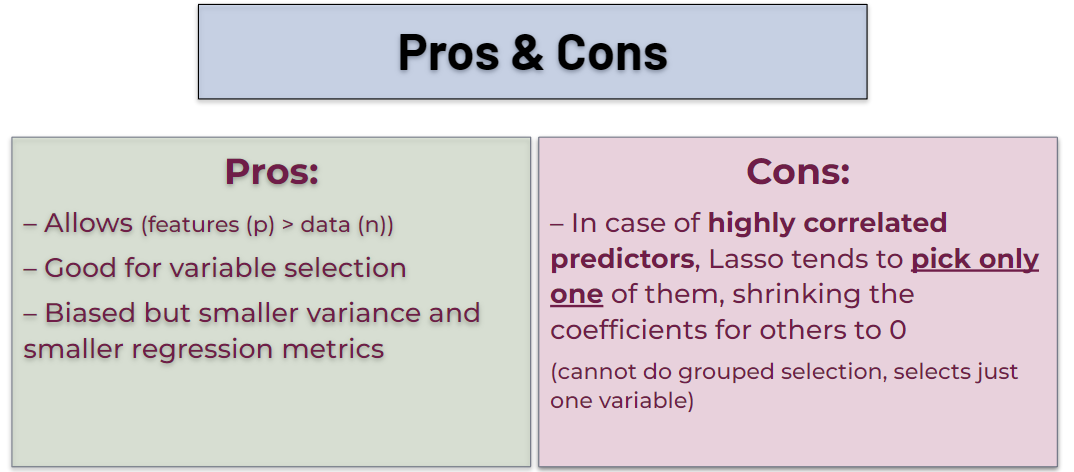
Ancak, bazen bazı independent değişkenlerin (featureların) tamamen elenmesi gerekmeyebilir. Bu nedenle, LASSO'nun selective feature'lara dikkat diyoruz!!

Yani **Multicollinear** durumda lassoyu kullanırken dikkatli olun, birisinin kafasını uçururken seçici olmaz.  Bu yüzden domain bilginize istinaden sizin arasındaki seçimi kontrol etmeniz gerekir. Prediction açısından değişiklik olmasada feature selection için önemli bir durum.

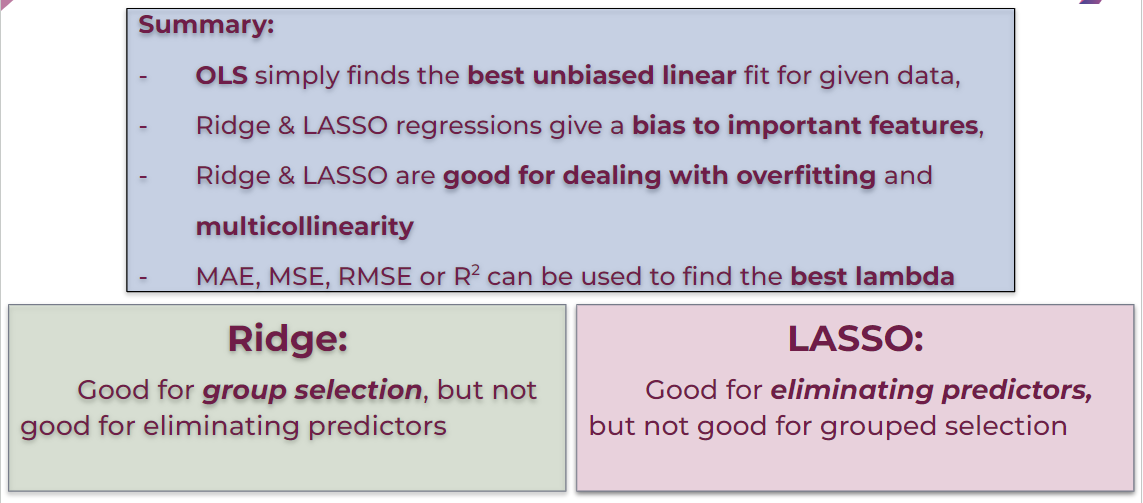
LASSO kullanırken aşağıdaki konulara dikkat etmek önemlidir:

1. Ölçeklendirme: LASSO, verilerin ölçeğine duyarlıdır. Bu nedenle, verilerin ölçeğini birbirine yakın hale getirmek veya normalleştirmek, modelin performansını arttırmaya yardımcı olabilir.
2. Katsayıların Sıfıra Yaklaşması: LASSO, bir düzenleme terimi kullanarak katsayıları sıfıra yaklaştırır. Bu özellik, modeldeki gereksiz bağımsız değişkenleri ortadan kaldırarak modelin genel performansını arttırmaya yardımcı olur. Ancak, bazen bazı bağımsız değişkenlerin tamamen elenmesi gerekmeyebilir. Bu nedenle, LASSO'nun seçici özelliklerine dikkat edilmelidir.
3. Lambda Değerinin Seçimi: LASSO'da, lambda adı verilen düzenleme terimi için uygun bir değer seçmek önemlidir. Lambda değeri, aşırı öğrenme ve modele uygunluk arasındaki dengeyi belirler. İdeal lambda değeri, çapraz doğrulama yöntemleri kullanılarak belirlenebilir.
4. Çoklu Doğrusal Bağlantı Sorunu: LASSO, veri setindeki çoklu doğrusal bağlantı sorununu ele alır. Ancak, bazı durumlarda bağımsız değişkenler arasındaki korelasyon çok yüksek olduğunda, LASSO'nun seçici özellikleri bozulabilir. Bu durumlarda, Ridge regresyonu daha iyi bir seçenek olabilir.
5. Eğitim Verisi Boyutu: LASSO'nun seçici özellikleri nedeniyle, eğitim verisi boyutu küçük olduğunda, LASSO'nun performansı düşük olabilir. Bu nedenle, LASSO'nun performansını arttırmak için daha büyük bir eğitim verisi kullanmak gerekebilir.

**LASSO regresyonun avantaj/dezavantajları:**



**Ridge – LASSO Karşılaştırması:**

****

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Alpha**, formüldeki lambdaya karşılık gelir.

**l1\_ratio** ise elastic net içinde ridge ve lasso oranını ayarlar. Başka bir hyperparametredir.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Feature Scaling :**

* ***Feature Scaling***
* ***Cross Validation and Grid Search***

**Scaling neden önemli:**

1. Gradient Descent algortihmasinin verimli ve hızlı çalışabilmesi için scaling önemli. zaman kazandırır.
2. Model coefficients: scale yapıldıysa rahatlıkla en yüksek coef olan feature en önemli etkiye sahip diyebiliriz. scaling yapılmadıysa söylenemez. Bu net.
3. Distance base lagoritmalar (mesafeye dayalı çalışan algoritmaların iyi çalısabilmesi için mutlaka scaling yapmalı.

Başka bir bakış açısı ile ML'de scaling yapmamızın üç önemli nedeni şunlardır:

1. Özelliklerin ölçeği arasındaki farklılıklar: Özelliklerin ölçeği arasındaki büyük farklılıklar, bazı özelliklerin diğerlerinden daha fazla ağırlığa sahip olmasına neden olabilir. Bu da modelin yanıltıcı sonuçlar üretmesine sebep olabilir. Özellikleri ölçeklendirmek, bu farklılıkları ortadan kaldırarak modelin daha doğru sonuçlar üretmesine yardımcı olur.
2. Optimizasyon algoritmalarının etkinliği: Bazı optimizasyon algoritmaları, özelliklerin ölçeklendirilmediği durumlarda yavaş veya hiç çalışmayabilir. Bu, özellikle büyük veri kümeleri veya karmaşık modellerde problem oluşturabilir. Özellikleri ölçeklendirmek, optimizasyon algoritmalarının daha hızlı ve verimli çalışmasına yardımcı olabilir.
3. Model performansının artırılması: Özellikleri ölçeklendirmek, model performansını artırabilir. Özellikle doğrusal regresyon, lojistik regresyon ve destek vektör makineleri gibi algoritmalar için ölçeklendirme yapmak, modelin daha doğru sonuçlar üretmesine yardımcı olabilir. Ayrıca, özellikleri ölçeklendirmek, aşırı uyumu (overfitting) azaltarak, modelin daha genelleştirilebilir hale getirilmesine de yardımcı olabilir

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

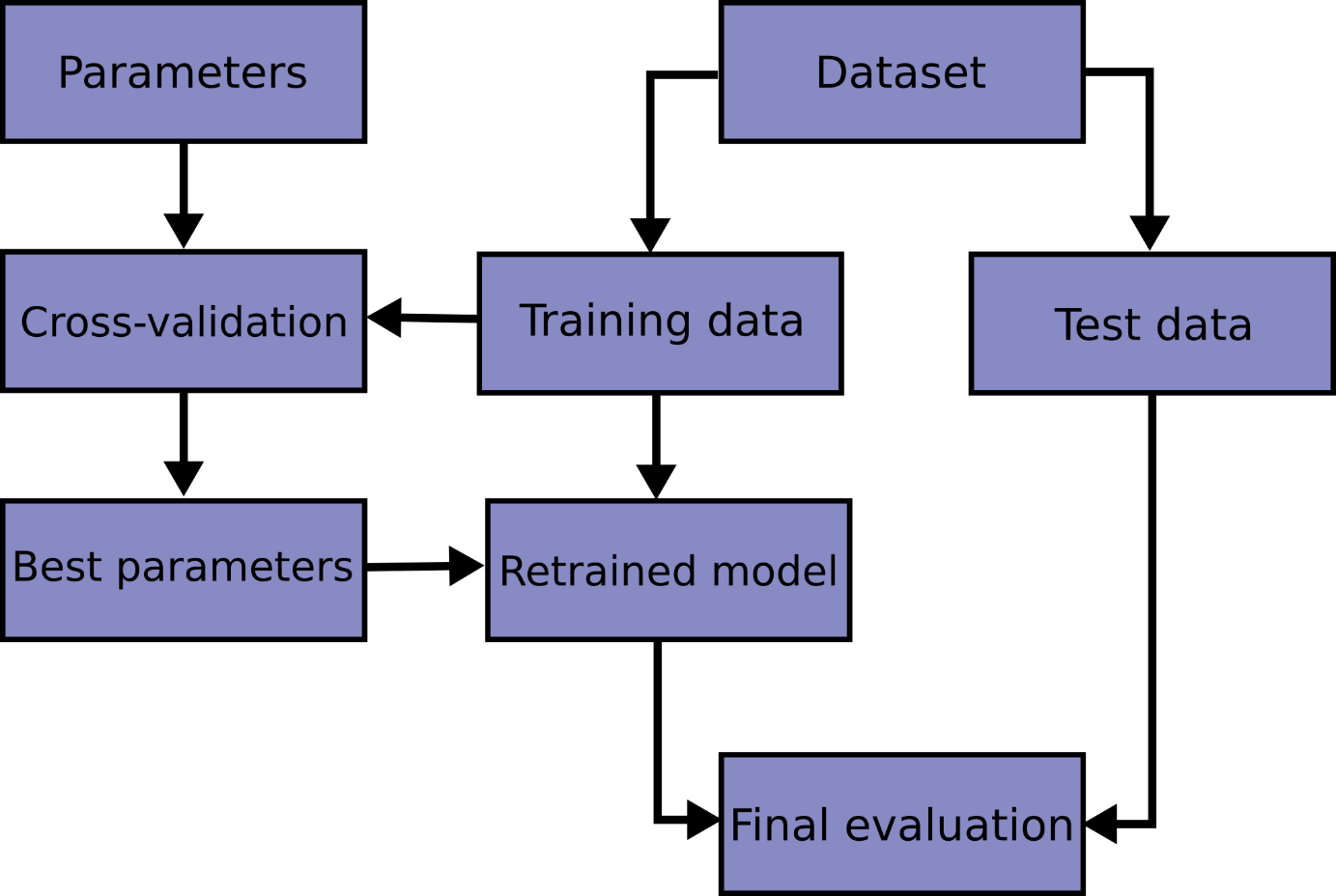
Eğer modelinizi ölçeklenmiş verilerle eğitiyorsanız, tahmin yapmadan önce gözlemlenen girdileri de ölçeklemelisiniz.

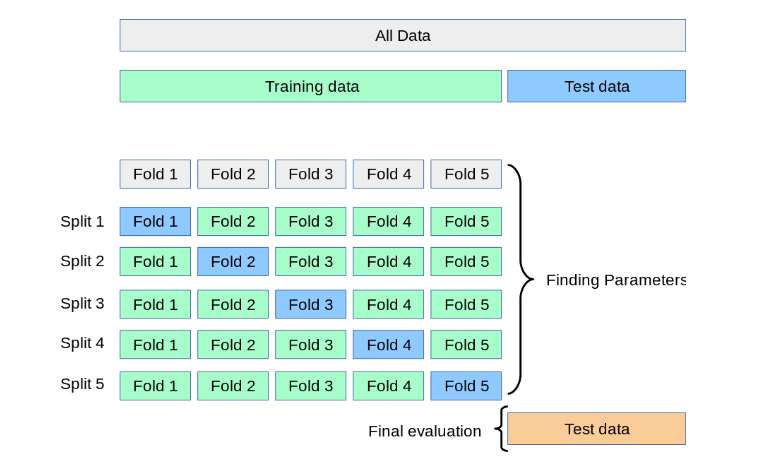
Bu, modelin featurelar arasındaki ölçek farklarını dikkate alarak doğru tahminler yapmasını sağlar. Yoksa modelin tahminleri yanıltıcı olabilir ve model performansı düşebilir. Bu yüzden modelin kullanımı için dışarıdan yeni veriler girilmeden önce bu verilerin de ölçeklendirilmesi gerekiyor.

Ölçeklenmiş katsayıları kolayca karşılaştırabilirsiniz. Ancak, bunları ölçeklenmemiş bir katsayı ile karşılaştıramazsınız.

Ölçeklendirmenin iyi performans gösterip göstermeyeceğinden emin değilseniz, ölçeklendirme yapmayı tercih edin.

**Cross Validation and Grid Search**





**Cross-validation;** bir modelin performansını ölçmek için kullanılır. Bu yöntem, **datasetini kümeler halinde bölerek her bir kümeyi sırayla test seti olarak kullanır ve geri kalan kümeleri eğitim seti olarak kullanarak modeli eğitir**. Bu işlem kümelerin tümü üzerinde tekrarlanır ve her seferinde farklı bir kümeyi test seti olarak kullanır. Bu sayede modelin genelleme performansı daha güvenilir bir şekilde ölçülebilir

**Grid search;** bir modelin **hiperparametrelerini** belirlemek için kullanılır. Bu yöntem, bir hiperparametre kümesi belirler ve bu kümedeki tüm kombinasyonlarını dener. Daha sonra her bir hiperparametre kombinasyonu için çapraz doğrulama yöntemi kullanılarak performans ölçülür ve **en iyi performansı veren hiperparametre kombinasyonu seçilir**. Bu yöntem, modelin performansını artırmak için en iyi hiperparametrelerin seçilmesine yardımcı olur

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Data leakage model performansınızda yalancı bir iyileşmeye neden olur. Bu da istenmeyen bir durumdur.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Scale ederken kullandığımız fit attribute'u ile scaling formülünü oluşturuyoruz. transform attribute'u ile de oluşturduğumuz formül üzerinden değerlerimizi dönüştürüyoruz.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fit\_transform attribute'unu kullandığımızda ise önce dönüşüm formülünü oluşturuyor, sonra dönüşümü yapıyor.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ML algoritmalarında iş akışımız genelde şu şekilde olacak:**

1. Sptlit the data.
2. Feature Scaling.
3. Modelleme
4. Cross Validation, Grid Search
5. Evaluation