Metrikler ve Kalite Nitelikleri Arasındaki Nedensellik İlişkisi

- **Hatırlatma:** Metriklerin temel kullanım amacı yazılım projelerindeki belirsizlikleri mümkün olduğu kadar giderip çeşitli öngörülerde bulunmak, **kararlar verebilmek** ve riskleri gidermektir.
  Örneğin; hangi birimlerde hata çıkma olasılığı yüksektir, bu sürüm kullanıma hazır mıdır, kusurlu birim sayısını ne kadar sürede belli bir sınırın altına indirebiliriz.
- Bu tür kararları verebilmek için alt düzey metrikler ile üst düzey kararlar (kalite) arasındaki ilişki (model) doğru kurulmalıdır.
- Eksik veya hatalı metrik kullanımı yanlış kararlara neden olabilir.

Örnek:

- Bir yazılım firması, sahadaki "başarılı" yazılımlarını belirleyip bunların geliştirme süreçlerinden dersler çıkarmak istiyor.
- Sahadan gelen hata bildirimlerine göre "hata sayısı / KLOC (bin kod satırı)" oranı düşük olan yazılım sistemleri "başarılı" örnek projeler olarak seçiliyor.
- Ancak daha sonra yapılan araştırmada, kullanıcılar memnun kalmadığı için bazı ürünlerin çok az kullanıldığı, bu nedenle bu sistemlerden az hata bildirimi geldiği anlaşılıyor.
- Buradaki sorun, başarı ölçmek için sadece "hata sayısı ve KLOC" metriklerinin kullanılmış olması. Yazılımın kullanım durumu da (sıklığı, süresi) dikkate alınmalıydı.

Metriklerin ve istatistiğin yanlış kullanımına örnekler

Örnek 1: Nedensellik ilişkisinin doğru kurulmaması

Gözlem:

- ABD'de yapılan bir araştırmada hava sıcaklığı ile trafik kazaları arasında pozitif korelasyon olduğu görülüyor. Hava sıcaklığı yükseldikçe kaza sayısı artıyor.

Hatalı yorumlar:

- Hava sıcaklığına bağlı olarak kaza sayısını öngörmek için bir model kurulabilir.
- Ancak bu model kaza nedenlerinin anlaşılmasında ve risklerin azaltılmasında yararlı olmaz.
  "Kazaları azaltmak için soğuk havalarda yolculuk yapın" veya "kazaları azaltmak için havayı soğutun" gibi geçerli önerilerde bulunulamaz.

Gerçek:

Hava sıcaklığı ile kaza sayısı arasında korelasyon olması sebep sonuç ilişkisi olduğunu göstermez.

Hava sıcaklığını ve kaza riskini artıran ortak nedenler (mevsim) olabilir. Örneğin
İyi havalarda sürücüler daha çok yol gidiyorlar (yolculuk süresi).
İyi havalarda sürücüler daha hızlı gidiyorlar (araç hızı).
İyi havalarda hava sıcaklığı daha yüksek oluyor.

Burada modeli sadece hava sıcaklığına göre değil, yolculuk süresi, hız gibi metrik-lere göre kurmak daha gerçekçi ve riskleri azaltmak için daha yararlı olacaktır.

Örnek 2: Nedenselliğin ters veya çift yönlü olması

Gözlem:

- Bir yazılım projesi süresince yazılım modüllerindeki değişimleri (satır değiştirme, ekleme ve çıkarma) ölçmek için evrimsel metrikler (*code churn metric*) kullanılır.
- Projesi süresince çok değişen yazılım sınıfları testlerde daha çok hata çıkartmaktadır.

Hatalı yorumlar:

- "Hataları azaltmak için sınıfları değiştirmeyin" gibi geçerli öneride bulunulamaz.

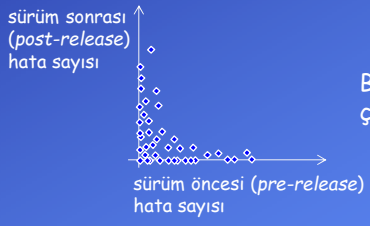
Gerçek:

- Hatalar ile değişim arasındaki sebep sonuç ilişkisi çift yönlü olabilir.
  - Hatalı sınıflar düzeltme amaçlı çok sık değiştiriliyorlardır. Değişim sebep değil, hatanın sonucudur.
  - Sınıflar çok değiştikleri için hata üretiyorlardır. Yazılım yapılan her etki hata çıkarma riski taşır.
- Değişim metrikleri (*code churn metric*) ilerideki hataları öngörmek için kullanılabilirler ancak modeller başka metriklerle de desteklenmeli.
- Eğer değişimler tasarım kusurlarını giderecek şekilde yapılırsa değişen sınıfların hata çıkarma olasılıkları da zaman içinde azalabilir.

Örnek 3: Etkili değişkenlerin gözden kaçması

Gözlem ve hatalı yorum:

- Bir yazılım projesinde ürün sahaya sürülmeden önce (*pre-release*) yapılan testlerde çok hata çıkaran modüllerin, yazılım sahada kullanılırken (*post-release*) diğer modüllere göre daha çok hata üretmesi beklenir.
- Bu beklentiye göre, sahada çalışan modüllerin kalitesi, sürüm öncesi testlerde çıkan hata sayısı ile orantılı olduğu düşünülür.
- Ancak gerçek projelerde farklı sonuçlar gözlemlenebilmektedir.
- Aşağıdaki diyagramda gerçek yazılımlardan toplanan verilere göre sürüm öncesi ve sonrası modüllerde çıkan hata sayıları gösterilmiştir. Diyagramdaki her nokta bir modüle (sınıf, metot, dosya) karşı düşmektedir.



Beklentinin tam tersi olarak sürüm öncesi çok hata çıkaran modüller sahada az hata üretebilirler.

Örnek 3: Etkili değişkenlerin gözden kaçması (devamı)

Gerçek:

- Beklentinin tam tersi olarak sürüm öncesi çok hata çıkaran modüller sahada az hata üretebilirler.
- Bunun bir nedeni **test kalitesinin** veya yoğunluğunun dikkate alınmamış olmasıdır.
  - Bazı modüller sürüm öncesi daha yoğun olarak test edildikleri için hatalardan arındırılmış olabilirler.
  - Bazı modüller ise sürüm öncesi yeteri kadar test edilmedikleri için az hata üretmişlerdir. Ancak bu modüller daha sonra sahada çok hata üretmişlerdir.
- Diğer gözden kaçan bir değişken de modüllerin **sahada kullanılma yoğunluğu**.
  - Programın bazı modülleri sahada çok kullanılmıyor, bu nedenle de hataları ortaya çıkmıyor olabilir.

Burada **test kalitesi** ve **kullanılma yoğunluğu** sonuçları etkileyen **gizli değişken**lerdir (***lurking variable***).

Örnek 4: Dengesiz veri kümeleri

- Bir projede modül boyutunun testlerde hata çıkma olasılığını etkilediğini düşünüyoruz.
- Kullanılan ölçüt: modül boyutu > 200 satır ise modül büyüktür.
- Şimdiye kadar hata çıkaran modüllerin %80'inde LOC>200 olduğu görülüyor.
- Projeden rastgele bir modül seçtiğinizde LOC>200 ise bu modülü nasıl sınıflandırırsınız (hatalı, hatasız)?

Çözüm için ek verilere gerek var:

- Bu projede test edilen modüllerde hata çıkma oranı %5 = 0.05.
- Hata çıkarmayan modüllerin %38'inde LOC>200 olduğu görülüyor.

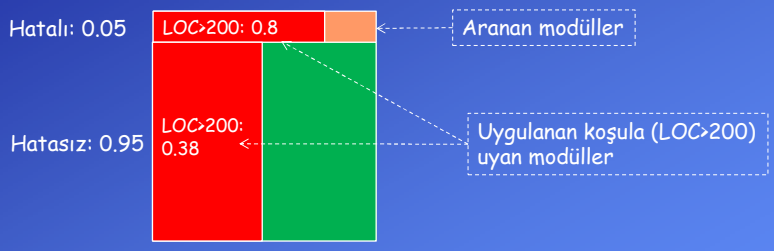
	Hatalı	:	Hatasız
Oran:	0.05	:	0.95
LOC>200:	0.8	:	0.38
LOC>200 ve Hatalı/Hatasız:	0.04	:	0.36
	1	:	9

Sonuç olarak bu projede LOC>200 olan bir modülün hata çıkarmama olasılığı, hata çıkarma olasılığından yaklaşık olarak 9 kat fazladır.

Örnek 4: Dengesiz veri kümeleri (devamı)

Analiz:

- Bu örnekte hatalı modül oranı hatasızlara göre çok düşüktür (dengesiz küme).
- LOC hataları belirlemek için uygun bir metrik olmakla beraber hatasız modüller kümesindeki LOC>200 olan modül sayısı, hatalı modüller kümesindekinden fazladır.
- Sadece LOC>200 koşulu ile yapılan bir sınıflandırma yanıltıcı sonuçlar vermektedir.



Öyle bir koşul seçilmeli ki

- Bu koşula uyan modüllerin büyük çoğunluğu hatalı olmalı (true positive/false positive yüksek olmalı).
- Koşula uyan hatasız sayısı (*flase positive*) az olmalı

Simpson Çelişkisi (*Simpson's Paradox*)

- Bir veri kümesi üzerinde yapılan analizden elde edilen sonuç, o veri gruplara ayrıldığında değişiyorsa buna Simpson çelişkisi denir.
- Kısaca, bir veriyi bir bütün olarak analiz etmekle gruplara ayırarak analiz etmenin farklı (çelişkili) sonuçlar üretmesidir.
- Bu durum karar vermede sorun yaratır. Hangi sonuç doğru ve geçerlidir?

Simpson, Edward H. (1951). "The Interpretation of Interaction in Contingency Tables". Journal of the Royal Statistical Society, Series B. 13: 238–241.

Daha önce Karl Pearson (1899) ve Udny Yule (1903) tarafından da oraya konmuş ancak daha sonraki yazılarda Simpson'ın makalesine referans verildiği için adı bu şekilde kalmış.

Örnek 1:

Gözlem:

- 1973 yılında UC Berkeley'de yüksek lisansa başvuran erkek adaylardan %44'ü kabul alırken kadın adaylardan %35'i kabul ediliyor.
- Kadın adaylar daha mı az başarılı?
- Üniversite erkek öğrencileri mi tercih ediyor?

Simpson Çelişkisi (*Simpson's Paradox*)
Örnek 1 (devamı):

- UC Berkley İstatistik Bölümü öğretim üyesi **Peter Bickel** verileri inceliyor.
- Kabul oranları bölümlere göre ayrı ayrı incelendiğinde bölümlerin çoğunda kadın adayların kabul oranlarının daha yüksek olduğu ortaya çıkıyor.
- En büyük altı bölümün dördünde kadın adaylar daha başarılı olmuş.

Bölüm	Toplam Başvuru	Toplam Kabul	Erkek Başvuru	Erkek Kabul	Kadın Başvuru	Kadın Kabul
A	933	64%	825	62%	108	**82%**
B	585	63%	560	63%	25	**68%**
C	918	35%	325	**37%**	593	34%
D	792	34%	417	33%	375	**35%**
E	584	25%	191	**28%**	393	24%
F	714	6%	373	6%	341	**7%**
Toplam	4526	39%	2691	**45%**	1835	30%

Örnek 1 (devamı):

Gerçek:

- Çelişkinin nedeni, çok sayıda kadın adayın kabul oranı daha düşük olan bölümlere başvurması.
- Çok sayıda erkek aday ise kabul oranı daha yüksek olan bölümlere başvurmuştur.
- Özellikle A, B, C ve E bölümlerindeki başvurulardaki dengesizlik ve kabul oranlarındaki farklılık çelişkinin oluşmasında rol oynamıştır.

Model oluştururken dikkat edilmesi gereken noktalar:

- Burada modellemeyi sadece kabul oranı ile yapmak doğru değildir.
- Modelde bölümlerin kabul oranları ve adayların başvuru sayıları da yer almalıdır.
- Sonuç olarak kadın adayların daha az başarılı oldukları doğru değildir.
- Bölümlere eşit (veya yakın sayıda) kadın ve erkek aday başvursaydı bölümlerin kabul politikası değişmeyecektiyse kadın adayların daha başarılı olduğu söylenebilir.
- Sonuç (bağımlı değişken) üzerinde etkili olduğu halde modelde yer almayan değişkenlere gizli değişken (*lurking variabale*) denir.
- Burada bölümlerin kabul oranları ve adayların başvuru sayıları gizli değişkenlerdir.

Simpson Çelişkisi (*Simpson's Paradox*)

Örnek 2:

Gözlem:

- Böbrek taşlarının tedavisinde kullanılan iki yöntemin (A ve B) etkinliği inceleniyor ve sonuçlar tıbbi bir makalede yayımlanıyor.*
- Her iki yöntem için de 350 olay inceleniyor ve aşağıdaki başarı değerleri elde ediliyor.

Yöntem	Olay	Başarılı tedavi	Başarı oranı
A	350	273	%78
B	350	289	%83

- Bu sonuçlara göre B yöntemi daha başarılı görünüyor.
- Böbrek taşı olan bir kişi B yöntemini mi tercih etmelidir?

* C. R. Charig, D. R. Webb, S. R. Payne, and J E Wickham, "Comparison of treatment of renal calculi by open surgery, percutaneous nephrolithotomy, and extracorporeal shockwave lithotripsy", Br Med J (Clin Res Ed). 1986.
https://doi.org/10.1136/bmj.292.6524.879

Örnek 2 (devamı):

Gerçek:

Olaylar böbrek taşlarının büyüklüğüne göre iki gruplara ayrılarak inceleniyor.

Yöntem / Taş boyutu	A	B
Küçük	Grup 1 **(81/87) %93**	Grup 2 (234/270) %87
Büyük	Grup 3 **(192/263) %73**	Grup 4 (55/80) %69
Toplam	(273/350) %78	**(289/350) %83**

- Yöntem A açık ameliyat uygulanan bir yöntemdir ve çoğunlukla daha zor olan büyük taşlarla ilgili olaylarda kullanılıyor.
- Yöntem B kapalı ameliyat uygulanan bir yöntemdir ve çoğunlukla daha kolay (başarı oranı yüksek) olan küçük taşlarla ilgili olaylarda kullanılıyor.
- Çok sayıda olayın yer aldığı Grup 2 ve Grup 3 toplam sonuçları etkiliyor, B yöntemi daha başarılıymış gibi görünüyor. Karşılaştırılan olaylar aynı (adil) değil.
- Burada taş büyüklüğü hem seçilen tedavi yöntemini hem de başarıyı etkileyen bir değişkendir. Bu şekilde hem bağımsız hem de bağımlı değişkeni (sonuç) etkileyen değişkenlere **yanıltıcı değişken** (***confounding variable***) denir.
- Bu tür değişkenleri dikkate almamak yanıltıcı sonuçlara neden olur.

Bayes Teoremi (Thomas Bayes, İngiliz matematikçi, 1702-1761)

Değerlendirmek istediğimiz bağımlı değişken (sonuç) ile bağımsız değişkenler olan metrikler (sebepler) arasında nedensel (*causal*) modeller kurmanın bir yöntemi de **Bayes ağlarıdır** (***Bayesian network***).

Bayes ağlarını açıklamak için önce koşullu olasılığı tanımlayan Bayes teorimi hatırlatılacaktır.

Koşullu olasılık:

Bayes teorimi, bir olay olduğunda ona bağlı başka bir olayın olma (koşullu) olasılığını hesaplamayı sağlar.

P(A|B): B olayının olduğu bilindiğine göre A olayının olma olasılığıdır.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

B'nin olma olasılığı P(B), eğer bilinmiyorsa bu değer koşullu olasılıklardan yararlanılarak aşağıdaki gibi hesaplanabilir.

$$P(B) = P(B|A) \times P(A) + P(B|notA) \times P(notA)$$

notA, A olayının olmamasını ifade eder, $P(notA) = P(A = False)$.

Örnek 1 (10.6'daki Örnek 4 ile aynı senaryo incelenmeketedir):

- Modül boyutunun hata çıkma olasılığını ne kadar etkilediğini belirlemek istiyoruz.
- **Sebep:** Modül boyutu > 200 satır (B) → **Sonuç:** Sınıfta hata çıkması (A)
- Bir modülün satır sayısının LOC>200 olduğunu biliyorsak o modülde hata çıkma olasılığı nedir? P(A|B) = ?
  - Bir projede test edilen modüllerde hata çıkma oranı %5, P(A) = 0.05.
  - Projede satır sayısı LOC>200 olan modül oranı %40 olarak belirleniyor.
  - Hata çıkaran modüllerin %80'inde LOC>200 olduğu görülüyor.

Çözüm:

- Satır sayısını bilmiyorsak hata çıkma olasılığı için öngörümüz P(A) = 0.05.
- LOC>200 olasılığı P(B)=0.4.
- Hata çıkaran modüllerin %80'inde LOC>200 olduğuna göre P(B|A) = 0.8.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} = \frac{0.8 \times 0.05}{0.4} = \mathbf{0.1} = \%\mathbf{10}$$

- Satır sayısını bilmediğimiz durumda hata çıkma olasılığı %5'ti.
- Modülün satır sayısının 200'den fazla olduğunu öğrendiğimiz anda hata çıkma olasılığı %10'a yükseldi.
- Projede LOC>200 olan modül oranı daha düşük olsaydı, bu olasılık daha da yükselecekti. Örneğin, P(B) = %10 → P(A|B) = %40.

Bayes Teoremi (devamı)

Örnek 2:

Bir hastalığı belirlemek üzere bir test geliştiriliyor.

- Eğer kişi hasta ise test onu mutlaka belirliyor. Hastalığı yakalama oranı %100. *False Negative = 0.*
- Kişi hasta değilse test hata yapabiliyor. Test olan sağlıklı kişilerin %5'inde hatalı olarak "hasta" sonucu veriyor. *False Positive = %5.*

Bin kişiden birinin hasta olduğu bir toplumda bu test uygulandığında sonucu pozitif çıkan birinin gerçekten hasta olma olasılığı nedir?

- Test sonucunu bilmiyorsak hasta olma olasılığı için öngörümüz P(A) = 0.001.
- Hasta olmama olasılığı için öngörümüz P(notA) = 0.999
- Hasta olan birinin testinin pozitif çıkma olasılığı P(B|A) = 1.
- Hasta olmayan birinin testinin pozitif çıkma olasılığı P(B|notA) = 0.05.
- Buna göre sonucu pozitif çıkan rastgele birinin gerçekten hasta olma olasılığı:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B|A) \times P(A) + P(B|notA) \times P(notA)} = \frac{1 \times 0.001}{1 \times 0.001 + 0.05 \times 0.999} = 0.0196$$

Test sonucu pozitif olmasına rağmen bir kişinin gerçekten hasta olma olasılığı %2'den düşüktür. Bunun nedeni, toplumda hasta oranının küçük olması ve buna da bağlı olarak testin keskinliğinin (*precision*) düşük olmasıdır.

Nedensel Modeller, Bayes Ağları
(*Causal Models, Bayesian Networks*)

- Bayes ağları, sebepler (koşullar, bağımsız değişkenler) ve sonuçlar (bağımlı değişkenler) arasındaki olasılık ilişkilerinin graf şeklinde gösterildiği yapılardır.
- Bir sebebe bağlı olan sonuç, başka bir sonucun sebebi olabilir.
  Örneğin; hava durumu → yolculuk süresi → kaza riski.
- Bir sonucu etkileyen birden fazla neden olabilir.
  Örneğin, kaza riski hem yolculuk süresine hem de araç hızına bağlıdır.
- Bir koşul birden fazla değişkeni etkileyebilir.
  Örneğin, hava durumu hem sıcaklığı hem de yolculuk süresini etkileyebilir.
- Birbirini etkileyen değişkenler birbirlerine bağlanarak bir ağ oluşturulur.
- Bir koşul gerçekleştiğinde ona bağlı değişkenin alacağı değerlerin olasılıkları bir tablo şeklinde yazılır.
- Bu tablolara düğüm olasılık tablosu (node probability table "**NPT**") veya koşullu olasılık tablosu (conditional probability table "**CPT**") denir.

Kaynak:
N. Fenton, M. Neil, *"Risk Assessment and Decision Analysis with Bayesian Networks", 2 ed., CRC Press, 2018.*

Örnek 1: Bir yazılım sınıfında hata olma olasılığı onun karmaşıklığına bağlıdır.

- Bu örnekte basitlik sağlamak için bağımsız (C) ve bağımlı (E) değişkenlerin ayrık (*discrete*) değerler (doğru/yanlış) aldığı varsayılmıştır.
- Karmaşıklık için bir eşik değeri belirlenebilir. Bunun üstündeki sınıflar "karmaşık" olarak sınıflandırılabilir.

Olasılık Tabloları:

1. "Sınıf karmaşık" P(C) tablosu:

	P(C)	
Karmaşık değil	Yanlış	0.9
Karmaşık	Doğru	0.1

Bu değerler projede inceleme yapılarak elde edilebilir.
Buna göre incelenen projede sınıfların %10'u karmaşıktır.

2. "Sınıf karmaşık (C) ise sınıfta hata var (E)" P(E|C) koşullu olasılık tablosu:

P(E|C)		Karmaşık?	
E \ C		Yanlış	Doğru
Hata yok	Yanlış	0.9	0.2
Hata var	Doğru	0.1	0.8

Bu değerler de incelenen projeden elde edilir.
Bu projedeki gözlemlere göre, karmaşık olmayan sınıfların %90'ında hata çıkmıyor, %10'unda hata çıkıyor.
Karmaşık sınıfların ise %80'inde hata çıkıyor.

Örnek 1 (devamı):

- Nedensel model kurulduktan sonra farklı senaryolara göre olayların olma olasılıkları aşağıdaki gibi hesaplanabilir.
  a) Bir sınıfta hata olmasının (koşulsuz) başlangıç olasılığı P(E).
  Bu aşamada sınıfın karmaşık olup olmadığı bilinmiyor. İki olası durum da $(P(C), P(notC))$ dikkate alınacak.

$$P(E) = P(E|C) \times P(C) + P(E|notC) \times P(notC)$$
$$= 0.8 \times 0.1 + 0.1 \times 0.9 = 0.17$$

Buna göre bu projede **rastgele** bir sınıfta hata olma olasılığı %17'dir.

b) Bir sınıfta hata bulduğumuzda bu sınıfın karmaşık olma olasılığını P(C|E) hesaplayabiliriz.
Bu bilgiyi sınıfla ilgili diğer değerlendirmelerde kullanabiliriz.

$$P(C|E) = \frac{P(E|C) \times P(C)}{P(E)} = \frac{0.8 \times 0.1}{0.17} = 0.47$$

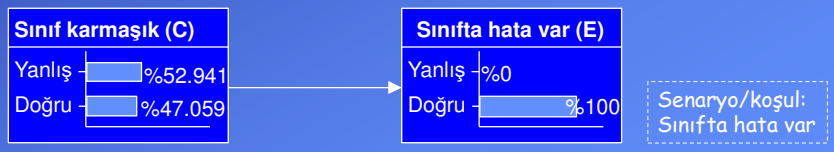
Projede karmaşık sınıfların genel oranı %10 olmasına rağmen eğer bir sınıfta hata buluyorsak o sınıfın karmaşık olma olasılığı %47'ye yükseliyor.

Örnek 1 (devamı):

- Bayes ağlarını kurmak ve farklı senaryolara (koşullara) göre incelemek için çeşitli yazılımlar bulunmaktadır.
- Bu yazılımlar belli bir koşul (veya birden fazla koşul) oluştuğunda değişkenlerin hangi değerleri alacağını görsel olarak da gösterirler.

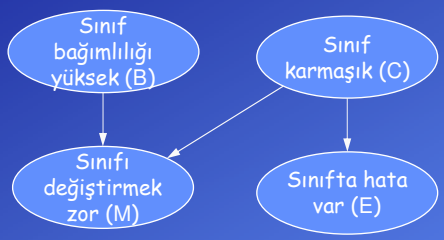
a) Modelin başlangıç (ön) (*prior*) durumu:

b) Modelin bilinen bir koşul (gözlem) gerçekleştikten sonraki (*posterior*) durumu:
Sınıfta hata olduğu (E = Doğru) biliniyorsa geriye dönük olarak sınıfın karmaşık olma olasılığı hesaplanabilir:



Örnek 2:

- Örnek 1'de sadece sınıfta hata olma olasılığı ile karmaşıklık arasındaki ilişki incelenmişti.
- Bu örnekte, sınıfların bağımlılığı (B) ve değiştirme zorluğu da (M) eklenerek model genişletilecektir.
- Yeni modele göre:
  - Bir yazılım sınıfında hata olma olasılığı (E) onun karmaşıklığına (C) bağlıdır.
  - Bir sınıfı güncelleme (değiştirme) zorluğu (M) ise o sınıfın karmaşıklığına (C) ve bağımlılığına (B) bağlıdır.



- Bu örnekte basitlik sağlamak için bağımsız ve bağımlı değişkenlerin ayrık (*discrete*) değerler (doğru/yanlış) aldığı varsayılmıştır.
- Bunu sağlamak için eşik değerleri kullanılabilir.
  - Örneğin CBO değeri belli bir eşik değerinden yüksek olan sınıflar bağımlılığı yüksek olarak değerlendirilebilir.
  - Sınıfı güncelleme süresi için de belli bir eşik değeri belirlenebilir. Bu süreden daha uzun sürede güncellenen sınıflar "zor güncellenen" olarak sınıflandırılabilir.

Örnek 2: Olasılık Tabloları

Örnek 1'deki olasılık tablolarına ek olarak aşağıdaki tabloları da oluşturmak gerekir.

1. "Sınıf bağımlığı yüksek" P(B)tablosu

		P(B)
Bağımlılık yüksek değil	Yanlış	0.6
Bağımlılık yüksek	Doğru	0.4

Buna göre incelenen projede sınıfların %40'ının bağımlılığı belirlenen eşik değerinin üstündedir.

2. "Sınıfı değiştirmek zordur" P(M | B, C) koşullu olasılık tablosu:
   - Sınıfı değiştirme zorluğu (M) iki bağımsız değişkenden (B ve C) etkilenmektedir.

P(M | B, C)	B	Yanlış		Doğru		
	C	Yanlış	Doğru	Yanlış	Doğru	
Sınıfı değiştirmek zor değil:	Yanlış	0.7	0.4	0.4	0.2	Sınıf çok bağımlı? / Sınıf karmaşık? / Sınıfı değiştirmek zor?
Sınıfı değiştirmek zor:	Doğru	0.3	0.6	0.6	0.8	

   - Bu değerler de incelenen projeden elde edilir.
   - Bu projedeki gözlemlere göre, bağımlığı yüksek olmayan ve karmaşık olmayan sınıfların %70'i kolay değiştirilebiliyor, %30'unu güncellemek zor oluyor.

Örnek 2: Modelin başlangıç durumu

- Örnek 1'de sınıfta hata olması olasılığının başlangıç değeri %17 olarak hesaplanmıştı.
- Gelişmiş modelde benzer şekilde sınıfı değiştirmenin zor olmasının olasılığının başlangıç değeri hesaplanacaktır.

Bir sınıfı değiştirmenin zor olmasının olasılığının P(M) başlangıç değeri:
Bu aşamada bağımlılık ve karmaşıklık koşulları bilinmiyor. Tüm olası durumlar dikkate alınacak.

$$
\begin{aligned}
P(M) = {} & P(M|B,C)P(B)P(C) && \text{Bağımlılık ve karmaşıklık yüksek} \\
& +P(M|B,notC)P(B)P(notC) && \text{Bağımlılık yüksek, karmaşıklık yüksek değil} \\
& +P(M|notB,C)P(notB)P(C) && \text{Bağımlılık yüksek değil, karmaşıklık yüksek} \\
& +P(M|notB,notC)P(notB)P(notC) && \text{İkisi de yüksek değil} \\
= {} & (0.8 \times 0.4 \times 0.1) + (0.6 \times 0.4 \times 0.9) + (0.6 \times 0.6 \times 0.1) + (0.3 \times 0.6 \times 0.9) \\
= {} & 0.446
\end{aligned}
$$

- Buna göre bu projedeki herhangi bir sınıfı değiştirmenin zor olmasının olasılığı yaklaşık %45'tir.

- Bayes teorimi çift yönlü çalıştığı için olasılıklar sonuçtan sebebe doğru da belirlenebilir.
- Modeldeki herhangi bir olasılık bilindiğinde diğer olayların olasılıkları buna bağlı olarak yeniden hesaplanabilir.

Örnek 2: Bilinen olasılıklara göre diğerlerinin hesaplanması

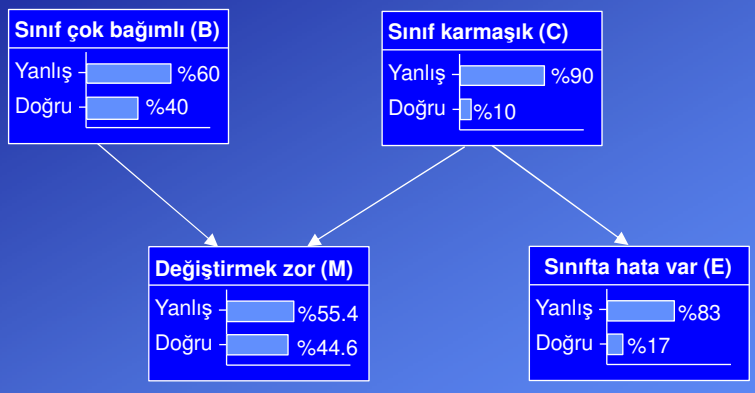
Örneğin bir sınıfta hata olduğu (E=True) biliniyorsa o sınıfı değiştirmenin zor olmasının olasılığı nasıl hesaplanır?

$$P(M|E) = P(M|B,C)P(B)P(C|E) \quad \text{Hata var, bağımlılık ve karmaşıklık yüksek}$$
$$+P(M|B,notC)P(B)P(notC|E) \quad \text{Hata var, bağımlılık yüksek, karmaşıklık yüksek değil}$$
$$+P(M|notB,C)P(notB)P(C|E) \quad \text{Hata var, bağımlılık yüksek değil, karmaşıklık yüksek}$$
$$+P(M|notB,notC)P(notB)P(notC|E) \quad \text{Hata var, bağımlılık yüksek değil, karmaşıklık yüksek değil}$$
$$= (0.8 \times 0.4 \times 0.47) + (0.6 \times 0.4 \times 0.53) + (0.6 \times 0.6 \times 0.47) + (0.3 \times 0.6 \times 0.53)$$
$$= 0.542$$

- Buna göre, bu projedeki hata çıkaran bir sınıfı değiştirmenin zor olmasının olasılığı yaklaşık %54'tir.
- P(M)'nin başlangıç değeri %45 iken bir sınıfta hata olduğunu bilmek bu olasılığı %54'e yükseltmiştir.

Örnek 2: Modelin farklı durumlara göre görselleştirilmesi

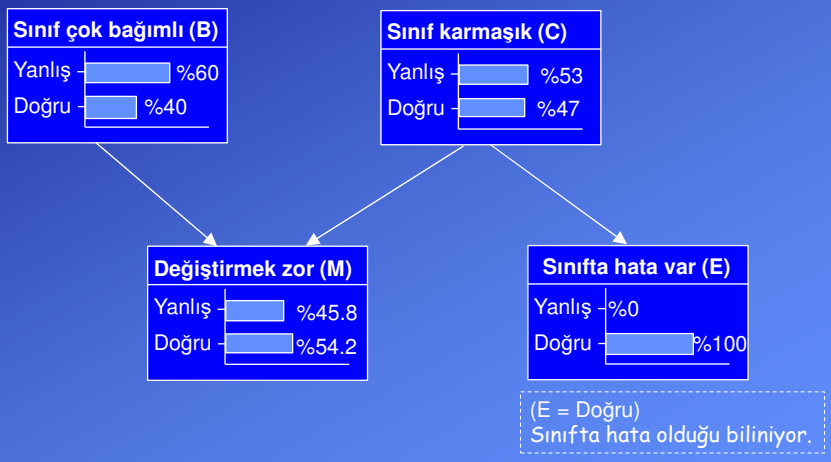
a) Modelin başlangıç (ön) (*prior*) durumu:



Örnek 2: Modelin farklı durumlara göre görselleştirilmesi

b) Modelin bilinen bir koşul (gözlem) gerçekleştikten sonraki (*posterior*) durumu:

Örneğin bir sınıfta hata olduğu (E = Doğru) biliniyorsa



(E = Doğru)
Sınıfta hata olduğu biliniyor.