MİRA AI

Teknik Rapor — Uygulanan Sistem, Yöntemler ve Ölçümler

Manipülatif İçerik ve Risk Analizi Sistemi

"Görmediğin etkiyi görünür kılar."

Alan	İçerik
Belge türü	Uygulanan sistemin teknik raporu — "ne yapıldı ve nasıl çalışıyor"
Önceki belge	"Teknik Ürün Gereksinimleri, Mimari ve Prototip Geliştirme Yol Haritası", v1.0 / 28 Temmuz 2026
Bu belgenin farkı	O belge "ne yapılacak" diyordu. Bu belge yapılanı, kullanılan yöntemi ve ölçülen sonucu anlatır.
Kapsam	Çekirdek risk motoru, Chrome eklentisi, metin modeli, görsel modeli — hepsi çalışır ve ölçülmüş durumda
Kapsam dışı	Android: rapor sonrası geliştirilen deneysel prototip (0.2.0-experimental); Chrome model metrikleri Android'e atfedilmez
Sürüm / Tarih	v1.1 / 24 Eylül 2026 (ilk baskı v1.0 / 7 Ağustos 2026)
Hazırlayan	Takım MİRA

İçindekiler

Word'de bu listeyi güncellemek için: listeye sağ tıkla → "Alanı güncelleştir" → "Tüm tabloyu güncelleştir".

İçindekiler ... 2
1. Ürün tanımı ve çözülen problem ... 4
1.1. Dört adımlı döngü ... 4
1.2. Çözülen problemler ... 4
2. Sistem mimarisi ... 4
2.1. Klasör yapısı ... 5
2.2. Sözleşmeler ... 5
3. Risk motoru ve açıklanabilir puanlama ... 5
3.1. Seviye eşikleri ... 5
3.2. Arayüz bileşeni içindeki ağırlıklar ... 6
3.3. İçerik bileşeni neden iki parçalı ... 6
3.4. Yapısal bir sınır ve çözümü ... 6
3.5. RiskEvent şeması ... 6
4. Arayüz dedektörleri ... 7
4.1. Ölçüm — fixture matrisi ... 7
4.2. Kendi kendini tespit sorunu ... 8
5. İçerik analizi — üç katman ... 8
5.1. Katmanlar neden ortalanmıyor, `max` alınıyor ... 8
5.2. İki ayrı taksonomi ... 8
5.3. Sözlük ağırlıkları ve maskeleme sınırı ... 8
5.4. Türkçe büyük İ sorunu ... 9
6. Platform adaptörleri ... 9
6.1. Tasarım ilkesi — CSS sınıf adına güvenme ... 9
6.2. Ölçüm — gerçek DOM kopyaları ... 9
6.3. Sanal liste sorunu ... 10
6.4. Mahremiyet sınırı ... 10
7. Müdahale katmanı ... 10
7.1. Üç kademeli tepki ... 10
7.2. Geri alınabilirlik zorunlu ... 10
7.3. Uyarı yorgunluğu kuralları ... 10
7.4. Müdahale yaşam döngüsü ... 11
8. Sürekli video analizi ... 11
8.1. Ölçüm — video karesi okuma maliyeti ... 11
9. Yapay zekâ — veri setleri ve model eğitimi ... 12
9.1. Metin modeli — TF-IDF + lojistik regresyon ... 12
9.2. Görsel modeli — MobileNetV3-Small transfer öğrenme ... 13
9.3. Ölçülen metrikler ve neden bunlar ... 15
10. Modelleri tarayıcıda çalıştırmak ... 15

10.1. Manifest V3 kısıtları ... 16
10.2. Ön işleme eşitliği — işin en zor kısmı ... 16
10.3. Hız — 58 ms'den 27 ms'ye ... 16
10.4. Tarayıcı doğrulaması ... 16
10.5. Kirlenmiş tuval ... 17
11. Mahremiyet ve güvenlik ... 17
11.1. Ölçüm — gönderi metni gerçekten yazılmıyor mu? ... 17
11.2. Tek bilinçli istisna ... 18
11.3. İzin yaklaşımı ... 18
12. Test ve doğrulama ... 18
12.1. Tarayıcısız testler ... 18
12.2. Tarayıcı doğrulama sayfaları ... 18
12.3. Uçtan uca sonuç ... 18
12.4. Derleme kapıları ... 19
12.5. Gerçek Instagram akışında doğrulama ... 19
13. Ölçümün yakaladığı hatalar ... 20
14. Bilinen sınırlar ve yol haritası ... 20
14.1. Henüz ölçülmemiş olan ... 20
14.2. Bilinen zayıflıklar ... 20
14.3. Yol haritası ... 21
Kaynaklar ve veri lisansları ... 21

1. Ürün tanımı ve çözülen problem

MİRA AI, sosyal medyadaki **manipülatif tasarım mekanizmalarını** ve **bağımlılığı özendiren içerikleri** birlikte analiz eden, bunların neden risk oluşturduğunu kanıtıyla açıklayan ve kullanıcıya doğru anda bilinçli seçim sunan bir sistemdir. Tüm analiz cihaz üstünde yapılır; sunucu yoktur.

Mevcut ekran süresi araçları çoğunlukla "ne kadar kullandın?" sorusunu cevaplar. MİRA AI **"neden durmak zorlaştı, hangi mekanizma etkiledi ve şu anda ne yapabilirsin?"** sorularını da cevaplar. Ürünün kullanıcıya söylediği cümle şuna benzer:

```
"Bu sayfada doğal bir bitiş noktası bulunmuyor. Son 30 saniyede 18 yeni
 içerik yüklendi. Bu nedenle durmayı fark etmek zorlaşabilir."
```

1.1. Dört adımlı döngü

Projenin özgünlük omurgası dört adımdan oluşuyor ve her adımın kodda bir karşılığı var:

Adım	Ne yapıyor	Nerede
Algıla	Altı arayüz dedektörü + üç katmanlı içerik analizi + davranış takibi	`core/detectors/`, `core/classifiers/`, `core/behavior/`
Açıkla	Her karar kanıt satırlarıyla birlikte üretilir; kanıtsız uyarı gösterilemez	`core/risk-engine/score.js`
Müdahale Et	Üç kademeli tepki ve 18 komut tipi; içeriği gizleyen her komut geri alınabilir	`core/commands.js`
Etkileşimi Kaydet	Hangi müdahalenin kabul edildiği kaydedilir ve sonraki seçimi etkiler. Etki ölçümü DEĞİL: kullanım süresinde ya da bağımlılık davranışında değişim ölçülmedi (pilot yok)	`core/risk-engine/intervention.js`

1.2. Çözülen problemler

- Sonsuz kaydırma ve otomatik oynatma doğal durma noktalarını ortadan kaldırabilir.
- Seri/streak, geri sayım ve rastgele ödül mekanizmaları tekrar kullanım baskısı oluşturabilir.
- Tütün, alkol ve kumar davranışlarını olumlu veya olağan gösteren içerikler normalleşme etkisi yaratabilir.
- Klasik uygulamalar yalnızca süre sınırı koyar; kullanıcının hangi mekanizmayla yönlendirildiğini açıklamaz.
- Herkese aynı uyarı zamanla görmezden gelinir; bağlama göre mikro müdahale gerekir.

> **Risk puanı klinik bir ölçek değildir**
> MİRA Risk Puanı, teknik davranışları karşılaştırmak için kullanılan **açıklanabilir bir ürün skorudur**. Tanı koymaz, sağlık değerlendirmesi yapmaz. Bu ayrım hem arayüzde hem bu belgede açıkça belirtilmek zorundadır.

2. Sistem mimarisi

Sistemin tek bir yapısal kararı var ve geri kalan her şey ondan türüyor: **`core/` karar verir, tüketici komutu çalıştırır.**

Bunun sebebi pratik. Kararı (eşik, cooldown, hangi müdahale) tüketiciye bırakmak, her tüketicinin aynı hesabı kendi başına tutarlı yapmasını beklemek demekti. Aradaki her küçük fark sessiz bir davranış farkı üretirdi —

aynı içerik tarayıcıda uyarı alırken telefonda almazdı ve bunu kimse fark etmezdi. Karar tek yerde toplandığında tüketicinin elinde yalnızca bir komut listesi kalıyor.



2.1. Klasör yapısı

Klasör	Sorumluluk
`core/`	Dedektörler, sınıflandırıcılar, risk motoru, komut katmanı. Platformdan bağımsız, tarayıcısız test edilebilir.
`extension/`	Chrome MV3 eklentisi. Karar vermez; `core/`'un ürettiği komutu DOM'a uygular.
`ai/`	Colab defterleri, veri boru hattı, model eğitimi ve değerlendirme betikleri.
`test-pages/`	Gerçek Instagram/YouTube DOM kopyaları ve tarayıcı doğrulama sayfaları.
`shared/`	Dondurulmuş sözleşmeler — RiskEvent şeması, özellik vektörü.
`docs/`	Ölçümler (`olcumler.md`), kurulum ve devir paketleri.
`tools/`	Yardımcı betikler: ORT vendorlama, statik sunucu, bu raporun üreticisi.

Çekirdeğin büyüklüğü: **25 modül, 4.865 satır.** Tarayıcı, ağ veya DOM gerektirmeden `node core/test/run.js` ile tamamı çalıştırılabiliyor.

2.2. Sözleşmeler

`shared/` altındaki şemalar dondurulmuş durumda: `RiskEvent` alanları, risk türü listesi ve özellik vektörü. Bir alan değişecekse sürüm numarası artıyor. Bu, iki ayrı tüketicinin aynı veriyi farklı yorumlamasını engelliyor.

3. Risk motoru ve açıklanabilir puanlama

```
MİRA Risk Puanı = 100 × (0,40 × Arayüz Riski
                       + 0,30 × İçerik Riski
                       + 0,30 × Davranış Riski)
```

Ağırlıklar bir başlangıç varsayımıdır ve pilot testlerle kalibre edilmelidir; "bilimsel olarak kesin oran" diye sunulamaz.

3.1. Seviye eşikleri

Puan	Seviye	Davranış
0–34	Düşük	Panelde bilgi; kesintisiz müdahale yok
35–64	Orta	Afiş veya açıklama kartı
65–100	Yüksek	Mikro müdahale kaplaması; “Devam et” seçeneği her zaman açık

3.2. Arayüz bileşeni içindeki ağırlıklar

Belge üç ana bileşenin ağırlığını veriyor ama her bileşenin içindeki dağılımı bize bırakıyor. Sıralama tek bir ilkeye dayanıyor: **bir mekanizma doğal durma noktasını ne kadar ortadan kaldırıyorsa o kadar ağır.** Sonsuz kaydırma durma noktasını tamamen yok eder; karanlık desen yalnızca tek bir kararı çarpıtır.

Mekanizma	Ağırlık
`INFINITE_SCROLL` — sonsuz kaydırma	0,35
`AUTOPLAY` — otomatik oynatma	0,25
`COUNTDOWN_FOMO` — geri sayım / aciliyet	0,15
`RANDOM_REWARD` — belirsiz ödül	0,10
`STREAK_PRESSURE` — seri kaybetme baskısı	0,10
`DARK_PATTERN` — asimetrik seçim tasarımı	0,05

3.3. İçerik bileşeni neden iki parçalı

```
İçerik riski = 0,7 × (bu öğenin riski) + 0,3 × (son bir dakikanın ortalaması)
```

Önce yalnızca pencerenin anlık görüntüsü kullanılıyordu ve ölçümde şu çıktı: **altı riskli gönderiden sonra gelen nötr gönderiler de 52 puan alıp “riskli içerik” etiketleniyordu** — pencerenin en yüksek değerini miras alıyorlardı. Yanlış alarm tam olarak böyle doğuyor. Formül ikiye ayrılınca karar ağırlıklı olarak elimizdeki gönderiye bakıyor, geçmiş yalnızca bağlam veriyor.

3.4. Yapısal bir sınır ve çözümü

İçerik ağırlığı 0,30 olduğu için içerik tek başına **30 puanı geçemiyor**, “orta” eşiği ise 35. Sonuç: yüzde yüz kesinlikle sigara içeren bir gönderi, sakin bir sayfada **yapısal olarak** yalnızca sessiz rozet alabiliyordu. Afiş hiç çıkamazdı.

Bu gerçek kullanımda yakalandı: YouTube’da “Sigara İkram Ediyor” başlıklı bir video sözlükte 0,75 risk aldı, bileşik skor 32’de kaldı, kullanıcı yalnızca rozet gördü. Oysa ürünün asıl iddiası tam da bu içeriği gerekçesiyle göstermek.

Çözüm: **oturum riski ile öğe bulgusu ayrı sorular.** Kanıtlı ve güvenli bir içerik bulgusu (`CONTENT_ALERT_MIN_RISK` = 0,70) oturum sakin olsa bile en az afiş hak ediyor. Kaplamaya yükseltilmiyor — kullanıcının önünü kesmek hâlâ oturum düzeyinde bir karar. Susturma ve yanlış tespit kuralları bu yükseltmeyi atlamıyor.

Bu taban kuralı kısaca **içerik ≥ 0,70 → skor ≥ 35** biçimindedir.

3.5. RiskEvent şeması

Her karar tek bir veri yapısına dönüşüyor. Serbest metin yok; `evidence` alanı sınıf etiketi taşıyor, gönderi metni değil (bkz. 11. bölüm).

```
{
  "event_id":   "uuid",
  "timestamp":  "2026-08-07T23:30:00+03:00",
  "source":     "chrome",
  "risk_type":  "RISKY_CONTENT",
  "severity":   72,          // 0-100
  "confidence": 0.94,
  "evidence":   ["Riskli içerik tespit edildi (Riskli davranışı normalleştirme)"],
  "components": { "arayuz": 0.80, "icerik": 0.82, "davranis": 0.00 },
  "intervention": "BLUR_CONTENT",
  "user_action":  "ACCEPTED | DISMISSED | FALSE_POSITIVE"
}
```

4. Arayüz dedektörleri

Altı dedektör, hepsi **kural tabanlı**. Bu bilinçli bir seçim: sonsuz kaydırmayı model ile tespit etmek hem daha az açıklanabilir hem daha az güvenilir olurdu. Kanıt cümlesi de ancak kuraldan çıkarılabiliyor — "30 saniyede 18 yeni kart eklendi" gibi bir gerekçeyi bir sınıflandırıcı üretemez.

Dedektör	Yöntem
Sonsuz kaydırma	`MutationObserver` ile yeni kart sayımı; sayfada sayfalama/bitiş işareti aranması; birim zamanda artan DOM düğümü
Otomatik oynatma	`play` olayı, `autoplay` niteliği, kullanıcı jesti olmadan başlayan medya
Geri sayım / FOMO	Zaman deseni taşıyan metin + saniyede değişen DOM içeriği
Seri / streak	Rozet ve seri metinleri, tekrar giriş baskısı kalıpları
Rastgele ödül	Çark, kutu, sürpriz ödül ve belirsiz sonuç bileşenleri
Karanlık desen	Kabul ve vazgeçme düğmelerinin görünürlük farkının ölçülmesi

4.1. Ölçüm — fixture matrisi

Her desen izole bir HTML sayfasında render edilip altı dedektör **birlikte** çalıştırıldı. Ölçülen şey iki yönlü: dedektör kendi desenini buluyor mu, **ve bulmaması gerekeni bulmuyor mu.**

Fixture	Beklenen	Tetiklenen	Sonuç
`none`	(hiçbiri)	(hiçbiri)	geçti
`infinite_scroll`	INFINITE_SCROLL	INFINITE_SCROLL	geçti
`autoplay`	AUTOPLAY	AUTOPLAY	geçti
`countdown`	COUNTDOWN_FOMO	COUNTDOWN_FOMO	geçti
`streak`	STREAK_PRESSURE	STREAK_PRESSURE	geçti
`random_reward`	RANDOM_REWARD	RANDOM_REWARD	geçti
`risky_content`	(hiçbiri)	(hiçbiri)	geçti
`dark_pattern`	DARK_PATTERN	DARK_PATTERN	geçti

Doğru: 6 · Kaçırılan: 0 · Yanlış pozitif: 0. `risky_content` satırında hiçbir arayüz dedektörünün tetiklenmemesi doğru davranıştır: içerik riski arayüz dedektörleriyle değil, metin ve görsel sınıflandırıcısıyla ölçülüyor.

4.2. Kendi kendini tespit sorunu

Matris ilk çalıştırıldığında `STREAK_PRESSURE` **her sayfada** tetikleniyordu; fixture sayfasının gezinme çubuğu desen adlarını görünür metin olarak taşıyordu. Test artefaktı gibi görünen bu durumun üretimde daha ciddi bir karşılığı var: **MİRA'nın kendi uyarı metni de sayfa metnine dahil.** Analiz dışı bırakılmazsa ürün kendi uyarısını manipülatif tasarım sanıp sonsuz geri besleme döngüsüne girer. `data-mira-ignore` niteliği bu yüzden eklendi ve MİRA'nın bütün arayüz bileşenleri onu taşımak zorunda.

5. İçerik analizi — üç katman

Katman	Rolü	Boyut	Yoksa ne olur
Sözlük	Birebir eşleşme, yüksek kesinlik	~5 KB	Her zaman var
TF-IDF metin modeli	Genelleme	2,88 MB	Sözlük çalışmaya devam eder
ONNX görsel modeli	Görsel sinyal	6,10 MB	Metin katmanları devam eder

> **Kademeli bozulma ilkesi**
> Hiçbir katman zorunlu değil. Model dosyası indirilemese, yavaş kalsa veya bozuk olsa bile ürün çalışmaya devam ediyor ve komut üretiyor. Tek bir dosyanın yokluğu bütün sistemi kör bırakmıyor.

5.1. Katmanlar neden ortalanmıyor, `max` alınıyor

Katmanlar birbirinin ortalaması alınacak rakip tahminler değil, **farklı kanıtlar.** Sözlük "deneme bonusu" ifadesini gördüyse bu kesin bir sinyaldir; görsel modelin emin olmaması onu zayıflatmamalı. Bu yüzden en güçlü kanıt kazanıyor.

Tersi de doğru: **negatif kontrol her zaman kazanır.** Bağımlılıkla mücadele, haber ve tedavi içerikleri riskli kelimeleri barındırır ama zararlı değildir. "Yeşilay", "bırakmak istiyorum", "zararları", "bağımlılık yapar", "alo 191" gibi ifadeler eşleştiğinde risk sıfırlanıyor ve diğer katmanlara hiç bakılmıyor. Bu liste olmadan ürün, Yeşilay'ın kendi paylaşımını riskli işaretlerdi.

5.2. İki ayrı taksonomi

Sözlük **madde** tespit ediyor (kumar, alkol, tütün); model ise **mekanizma** sınıfları üretiyor (FOMO, streak, rastgele ödül, normalleştirme). Bunlar farklı sorular: "bu içerik neyi anlatıyor" ve "bu içerik nasıl yönlendiriyor". Bir bahis reklamı hem `kumar` maddesi hem `NORMALIZING_RISK` mekanizmasıdır. Ana etiket mekanizma, `substance` ikincil alan olarak taşınıyor.

5.3. Sözlük ağırlıkları ve maskeleme sınırı

Sözlük eşleşmeleri farklı kesinlik taşıyor ve otomatik maskeleme sınırı bu farkı biliyor. Sınır 0,75'te ve ayrım kendiliğinden doğru yere düşüyor: **madde içeren içerik maskeleniyor, baskı dili yalnızca anlatılıyor.** Gönderinin içinde çalışan baskı bileşeni (işleyen geri sayım, ödüllü başlatan düğme) ise yapısal kanıt: yalnız o bölüm bulanıklaşıyor.

Kural	Ağırlık	Otomatik maskeleme
“deneme bonusu”, “çevrimsiz”, “yatırım şartsız”	1,00	evet
“bahis”, “iddaa”, “casino”, “canlı bahis”	0,85	evet
“sigara”, “nargile”, “vape”, “puro”	0,75	evet
Geri sayım / aciliyet kalıpları (metin)	0,70	hayır — metin tek başına yetmiyor
Gönderide işleyen geri sayım ya da ödül düğmesi (yapı)	—	evet — yalnız o bölüm
Seri kaybetme baskısı	0,70	hayır
“rakı”, “viski”, “bira”, “şarap”	0,60	hayır — yemek bağlamında da geçiyor

Metinde “son 2 saat” geçmesi bir haber de olabilir; gönderinin içinde saniye saniye azalan bir sayaç ise kanıtın kendisi. Sigara sahnesinde medya, kampanya gönderisinde yalnız sayaç bölümü bulanıklaşıyor; ikisi de tek dokunuşla açılıyor.

5.4. Türkçe büyük İ sorunu

JavaScript’te `/sigara/i` ifadesi **`"SİGARA"` metnini yakalamaz.** Türkçe büyük İ, Unicode küçültmesinde birleşen bir nokta bırakıyor ve karakter eşleşmiyor. Kumar ve tütün reklamları sosyal medyada tam olarak büyük harfle yazıldığı için bu, sessizce en riskli içerikleri kaçırmak demekti.

`foldTr()` eklendi: düz ASCII, küçük harf, aksansız katlama. Aynı işlev üç dilde birebir aynı sonucu vermek zorunda — JavaScript, Python (eğitim) ve Kotlin — ve bu testle korunuyor.

6. Platform adaptörleri

Dedektörler ve risk motoru hangi sitede çalıştıklarını bilmiyor; platforma özel her şey adaptörlerde toplanıyor. Instagram yarın DOM’unu değiştirirse tek dosya güncelleniyor.

6.1. Tasarım ilkesi — CSS sınıf adına güvenme

Instagram ve YouTube sınıf adlarını her derlemede yeniden üretiyor (`_ab12cd`). Semantik yapı ise stabil: `article`, `img`, `video`, `dir="auto"`, ARIA nitelikleri ve Polymer bileşen adları (`ytd-rich-item-renderer`). Adaptörler yalnızca bunlara bağlanıyor.

6.2. Ölçüm — gerçek DOM kopyaları

`test-pages/instagram.html` ve `test-pages/youtube.html` gerçek platformların DOM **yapısını** taklit ediyor (tasarımını değil) ve **gerçek adaptörle** analiz ediliyor. Kendi demo akışımızda test etmek hiçbir şey kanıtlamazdı; orada seçicileri de biz yazdık.

Ölçüm	Instagram kopyası	YouTube kopyası
Bulunan öğe / gerçek öğe	**11 / 11**	**14 / 14**
Başlığı çıkarılamayan	0	0
İç içe tekrar analiz	0	0
Başlıkta arayüz metni kirliliği	0	0

Son satır önemli: “Beğen / Yorum yap / Paylaş / Kaydet” ve beğeni sayıları başlıktan tamamen ayıklanıyor. Her gönderide aynı oldukları için model onları öğrenip gürültü üretirdi.

6.3. Sanal liste sorunu

YouTube Shorts sonraki videoda yeni bir DOM öğesi yaratmıyor; **aynı öğenin içeriğini değiştiriyor** ve DOM'da aynı anda üç reel tutuyor (önceki/aktif/sonraki). Öğe kimliğine göre tekilleme yapan kod bu yüzden ikinci videodan itibaren kör kalıyordu.

Çözüm: kimlik öğeden değil **içerikten** türetiliyor — sırasıyla bağlantıdaki video kimliği, küçük resim adresindeki kimlik, adaptöre özel kimlik, son çare öğe kimliği. Oynak metin (görüntülenme sayısı, "2 saat önce") imzaya **asla** girmiyor: girseydi imza her saniye değişir, öğe saniyede birkaç kez yeniden analiz edilir ve davranış bileşeni gerçek zamanın kat kat üstünde şişerdi.

Ayrıca `/shorts/` yolunda yalnızca **aktif** reel analiz ediliyor. Üçünü birden analiz etmek aynı video için üç ölçüm, komşuların başlığının yanlış içeriğe yazılması ve davranış bileşeninin üç katına çıkması demekti.

6.4. Mahremiyet sınırı

Instagram'ın `/direct/` (özel mesajlar) ve `/accounts/` yolları hiçbir koşulda analiz edilmiyor. Instagram tek sayfa uygulaması olduğu için bu kontrol **her URL değişiminde** tekrarlanıyor; bir kez başlatıp bırakmak, akıştan özel mesajlara geçildiğinde sınırın sessizce ihlali olurdu.

7. Müdahale katmanı

7.1. Üç kademeli tepki

Kademe	Ne zaman	Ne görünüyor
Sessiz	Düşük risk, sinyalsiz öğe	Yalnızca kalıcı oturum göstergesi
Afiş	Orta risk veya kanıtlı içerik bulgusu	Alt kısımda küçük öneri, 15 sn sonra kendiliğinden kapanıyor
Kart	Yüksek risk veya maskeleme uygulandı	Ortada açıklama kartı: bulgu, kanıt, seçenekler

Öğenin üstüne rozet basma davranışı kaldırıldı. Ana sayfada 20 kutucuğun 20'sinde rozet çıkması bilgilendirmiyor, kirletiyordu; ayrıca rozet kabuğu sitenin yapışkan başlığından yüksek bir katmanda durduğu için kaydırıldığında YouTube'un arama çubuğunun üstüne biniyordu.

7.2. Geri alınabilirlik zorunlu

İçeriği gizleyen **her** komut geri alınabilir olmak zorunda ve bu kod düzeyinde zorlanıyor: geri alma karşılığı olmayan ya da hedefi bulunmayan bir maskeleme komutu üretilemiyor, komut kurucusu hata fırlatıyor.

İkinci kural gerçek bir hatadan doğdu: Shorts'ta içerik `<video>` olduğu için görsel arayan kod `null` dönüyor, `BLUR` komutu hedefsiz üretiliyor ve çalıştırıcı sessizce çıkıyordu. Kullanıcı "Bulanık bırak"a basıyor, hiçbir şey olmuyor, hiçbir yerde iz kalmıyordu. Hedefsiz bir maskeleme komutu tanım gereği hiçbir şey yapmaz; **üretilebilmesi bile yanlış.**

Bu bir üslup tercihi değil: manipülatif tasarımla savaşan bir ürün kullanıcıyı içeriğin arkasında kilitleyemez.

7.3. Uyarı yorgunluğu kuralları

- Aynı risk türü için cooldown: davranış/arayüz uyarılarında **10 dakika**, kanıtlı içerik bulgusunda **90 saniye**.
- Kullanıcı üç kez üst üste "Devam et" derse o risk türü oturum boyunca susturuluyor.
- İki kez "bu tespit yanlış" denirse o risk türünde ısrar edilmiyor.
- Kullanıcıyı serbest bırakan seçenek daima listenin sonunda ve **diğerleriyle aynı boyutta** — sıralama onu öne çekerse ürün kendi kendini etkisizleştirir.

7.4. Müdahale yaşam döngüsü

Uygulanan her müdahale bir risk olayına **ve bir akış öğesine** bağlı. Öğenin içeriği değiştiğinde bulanıklık, duraklatma ve "Göster" düğmesi birlikte serbest bırakılıyor.

Bu kavram olmadan sanal listelerde doğrudan hataya dönüşüyordu: müdahaleler dört ayrı yerde tutuluyor ve hiçbiri "bu öğe artık başka bir içerik gösteriyor" olayına bağlı değildi. Kullanıcı bir kez "Bulanık bırak" dedikten sonra sonraki bütün videolar bulanık geliyordu.

Bir ayrıntı kritik: **bulanıklaştırılan nesne ile kimliği taşıyan nesne farklı.** Blur medyaya (`<video>`) uygulanıyor, kimlik ise akış öğesinde. Gerçek Shorts tek bir paylaşılan oynatıcı kullanıyor ve o öğe bütün oturum boyunca yaşıyor; maskeleme ona bağlandığında hiç ölmüyor ve nöbetçi her turda bulanıklığı geri koyuyordu.

8. Sürekli video analizi

Ürünün en ayırt edici parçası. Bir video akışında **tek kare bir videoyu temsil etmiyor**: riskli sahne videonun başında değil 12. saniyesinde çıkabilir. İlk sürümde görsel katmanı video başına en fazla bir kez çalışıyordu — üstelik çoğu zaman hiç, çünkü ilk ölçüm video daha tamponlanmadan yapılıyor ve hazır olmayan videodan kare okunamıyordu.

Artık oynayan video **1,5 saniyede bir** yeniden ölçülüyor, öğe başına en çok 40 örnek (yaklaşık 60 saniye). Bunu güvenli kılan üç kural var:

Kural	Neden gerekli
İçerik penceresi öğe kimliğiyle tekilleniyor	Yoksa 30 saniyelik tek bir video "son bir dakikada 20 riskli içerik geçti" diye raporlanır — uydurma bir kanıt ve şişmiş bir ortalama
Karedeki bulgu **iki ardışık örnekte** aynı sınıf olmalı	Tek karelik parlama müdahale üretmesin
Örnekleme aralığı kendini sınırlıyor: `süre / 0,1`	Analiz 300 ms sürdüyse sonraki örnek 3 saniye sonra alınır; yavaş makinede sayfa takılmaz
Yeniden ölçüm kaydırma sayılmıyor	Aynı videoyu 40 kez örneklemek 40 kaydırma sayılsaydı davranış bileşeni gerçek zamanın kat kat üstüne çıkardı

Onay, eşiğin işini daha iyi yapıyor. Tek karelik gürültüyü bastırmanın iki yolu var: eşiği yükseltmek ya da onay istemek. Eşik duyarlılığı da götürüyor, onay götürmüyor. Deneysel video-kare yolunda bu nedenle 0,80 eşiği tanımlı; ancak bu yol finalde varsayılan olarak **KAPALI**. Fotoğrafta ölçülmüş 0,90 çalışma noktası aynen duruyor.

8.1. Ölçüm — video karesi okuma maliyeti

Kare tuvale kaynağın doğal boyutunda okunduğunda (Shorts'ta 1080×1920) her örnekte 8 MB'lık bir GPU→CPU kopyası yapılıyor ve oynatıcı hırpalanıyor. Boru hattının ilk işi zaten kısa kenarı 256'ya indirmek olduğu için okuma o boyutta yapılıyor.

Ölçü	Değer
Video doğal boyutu	270×400
Tuvale okunan	**256×379**
Piksel yolu	`dogrudan`
Kare okuma	**4,8 ms**
Ön işleme (tensöre)	**7,9 ms**
Toplam (çıkarım dahil)	≈ **31 ms/örnek** — 1,5 sn aralıkta ana iş parçacığının %2'si

9. Yapay zekâ — veri setleri ve model eğitimi

Her probleme “AI” etiketi yapıştırılmadı. Hangi problemde kural tabanlı yöntemin daha doğru, hangi problemde modelin gerekli olduğu ayrıldı:

Problem	Yöntem	Gerekçe
Sonsuz kaydırma, otomatik oynatma	Kural tabanlı DOM analizi	Daha açıklanabilir ve güvenilir; kanıt cümlesi ancak kuraldan çıkar
FOMO / seri dili	TF-IDF + lojistik regresyon	Bağlam ve dil kalıpları gerekiyor
Riskli görsel içerik	MobileNetV3-Small transfer öğrenme	Görsel ürün ve tema sinyalleri
Davranış örüntüsü	İstatistiksel taban çizgisi	Kişiye göre değişen kullanım

9.1. Metin modeli — TF-IDF + lojistik regresyon

Belgenin istediği açıklanabilir taban çizgisi. **Saf Python ile yazıldı, scikit-learn kullanılmadı** — çünkü model tarayıcıda çalışmak zorunda ve eğitim ile çıkarımın tek bir belirtimi paylaşması gerekiyordu. Model karara hangi kelimenin katkı yaptığını söyleyebiliyor; “Neden gösterildi?” ekranı bunun üzerine kurulu.

Veri seti

Köken	Örnek	Ne veriyor
Sentetik Türkçe (takım yapımı)	6.000	Altı sınıfın tamamı
Princeton “Dark Patterns at Scale”	859	Gerçek, insan etiketli FOMO (İngilizce)
`Overfit-GM/turkish-toxic-language` (zehirsiz)	900	Gerçek Türkçe, sosyal medya kaydı
`winvoker/turkish-sentiment-analysis`	842	Gerçek Türkçe, ürün yorumları
Toplam	**8.601**	Gerçek veri oranı **%30,2**

Eğitim/doğrulama/test ayrımı **kaynak bazlı** yapıldı (%70,6 / %14,8 / %14,6), satır bazlı değil. Sebebi ölçümle görüldü: ilk veri setinde her kaynak tek bir cümle üretiyordu ve model şablonları ezberledi.

```
eğitim makro F1 = 1,0000    doğrulama = 0,63    test = 0,24
```

Satır bazında bölseydik ~0,95 görüp modeli iyi sanacaktık; kaynak bazlı ayrım onu görülmemiş şablonlarla sınayınca çöktü. Düzeltme: her kaynak artık birden çok farklı ifade taşıyor, böylece model cümleyi değil deseni öğrenmek zorunda.

Sonuç — ve gerçek verinin etkisi

Ölçüm	Yalnız sentetik	+ gerçek veri
Makro F1	0,971	**0,885**
Yanlış alarm	0,003	**0,000**

Gerçek veri skoru iyileştirmedi, dürüstleştirdi. İlk sayı yüksekti çünkü test kümesi de aynı üreteçten çıkıyordu; model şablon ailesini tanıyordu. Gerçek Türkçe metin eklenince "normal metin" tanımı genişledi ve görev gerçekten zorlaştı. Yanlış alarmın sıfıra inmesi bunun kanıtı.

Tek sayı iki gerçeği gizliyordu — dil kırılımı

Grup	n	Doğruluk	Makro F1
Türkçe	1.106	**0,942**	**0,939**
— sentetik Türkçe (6 sınıf)	610	0,895	0,899
— gerçek Türkçe (yalnız nötr)	496	1,000	—
İngilizce (Princeton)	151	0,377	0,548

0,939 ne ölçüyor, ne ölçmüyor. Türkçe testteki riskli sınıfların hepsi **sentetik** örneklerden geliyor; gerçek Türkçe test örneklerinin (496) tamamı nötr. Yani 0,939, modelin gerçek Türkçe sosyal medyadaki riskli içeriği yakalama başarısı değil; sentetik riskli + gerçek nötr karma kümedeki ayırma başarısıdır. Gerçek Türkçe riskli metinde başarım ölçülmedi.

Birleşik 0,885 bu yüzden tek başına yanıltıcı: hedeflenen Türkçede makro F1 0,939. İngilizce düşük çünkü eğitimin yalnızca %10'u İngilizce ve karakter n-gramları Türkçeye göre ayarlı. Akışta İngilizce reklam da geçtiği için kapsamda tutuluyor.

Sınıf bazında (test, n = 1.257)

Sınıf	Kesinlik	Duyarlılık	F1	n
NEUTRAL	0,787	1,000	0,881	567
FOMO_PRESSURE	0,975	0,545	0,699	211
STREAK_PRESSURE	1,000	0,951	0,975	82
RANDOM_REWARD	1,000	0,944	0,971	126
NORMALIZING_RISK	0,985	0,830	0,901	159
AD_DISCLOSURE_RISK	1,000	0,786	0,880	112

FOMO'nun düşük duyarlılığı doğrudan İngilizce sorunundan geliyor: test kümesindeki 211 FOMO örneğinin 151'i Princeton'dan, yani İngilizce.

Model boyutu: 2,88 MB JSON · **çıkarım:** < 5 ms · **bağımlılık:** yok · **eğitim:** yerel makine, CPU, ~2 dakika (GPU gerekmiyor).

9.2. Görsel modeli — MobileNetV3-Small transfer öğrenme

Alan	Değer
Mimari	MobileNetV3-Small, transfer öğrenme
Eğitim ortamı	Google Colab, T4 GPU — `ai/notebooks/02_gorsel_modeli.ipynb`
Sınıflar	NEUTRAL · TOBACCO · ALCOHOL · GAMBLING
Eğitim verisi	Open Images V7 (CC BY 4.0), insan doğrulamalı etiketler
Değerlendirme verisi	Open Images doğrulama + Mendeley Smoker Detection (CC BY 4.0) — **yalnızca test**
Gönderilen dosya	`gorsel_fp32.onnx`, 6,10 MB

Zor negatifler — yanlış alarm sandığımızın iki katı çıktı

İlk tarama Open Images'ın **kolay** negatifleriyle yapılmıştı: manzara, eşya, hayvan. Gerçek akış ise insan fotoğrafıyla dolu. Zor negatifler eklenince (inhaler kullanan, telefonla konuşan, su içen insanlar) aynı eşikte yanlış alarm **%4,5 → %9,4** çıktı. Test kümesi bu yüzden 968 görsele genişletildi ve çalışma noktası yeniden seçildi.

İki ayrı test kümesi — karıştırılmamalı

Görsel modeli iki farklı kümede ölçüldü ve **sayılar birbirinin yerine kullanılamaz.** Raporun ilk sürümünde bu ayrım yazılmamıştı ve iki makro F1 değeri aynı şeyi ölçüyormuş gibi görünüyordu.

	Defterin kendi bölümü	Zor test kümesi
Üreten	Colab defteri	`eval_image_threshold.py`
n	1.258	**968**
Bileşim	Open Images (eğitim bölümünden ayrılan)	Open Images **doğrulama** (744) + Mendeley `Testing/` (224)
Makro F1 (argmax)	0,631	**0,731**
TOBACCO F1	0,1935	0,5100
CLIP sıfır-atış	0,574	ölçülmedi

Fark kümeden geliyor: zor küme Mendeley'in `Testing/` bölümünü içerdiği için TOBACCO örneği 128'e çıkıyor; defterin bölümünde tütün örneği az ve o sınıfın F1'i gürültülü. **Ürünün çalışma noktası zor kümeden seçildi**, çünkü o küme gerçek akışa daha yakın.

İki eğitim, aynı zor test kümesinde (968 görsel)

Ölçüm	v1	v2 (gönderilen)
Makro F1 (argmax)	0,751	0,731
Yanlış alarm @ 0,90	0,0566	**0,0586**
Yanlış alarm @ 0,95	0,0352	**0,0293**
Duyarlılık @ 0,90	0,640	0,640

v2, argmax makro F1'de biraz geride ama **ürünün gerçekten kullandığı çalışma noktalarında daha iyi:** aynı duyarlılıkta daha az yanlış alarm. Ürün argmax kullanmıyor, dolayısıyla seçim v2 oldu. Bu ayrım önemli — tek bir özet metriğe bakarak seçim yapsaydık daha kötü modeli gönderirdik.

Seçilen çalışma noktaları

`argmax` ("en olası sınıf ne?") ürünün sorusu değil. Ürün **"işaretlemeye yetecek kadar emin miyiz?"** diye soruyor ve ölçü `1 − P(NEUTRAL)`: risk üç sınıfa bölünmüş olabilir ve önemli olan toplam risk.

Sabit	Değer	Yanlış alarm	Ne yapıyor
`IMAGE_MIN_RISK`	0,90	%5,86 (30/512 nötr)	Görsel katmanı devreye giriyor → afiş
`AUTO_MASK_MIN_RISK`	0,95	%2,93 (15/512 nötr)	Kullanıcıya **sormadan** bulanıklaştırma
`IMAGE_FRAME_MIN_RISK`	0,80	—	Deneysel video karesi; finalde varsayılan **KAPALI**

Sabit	Değer	Yanlış alarm	Ne yapıyor
`AUTO_MASK_MIN_RISK_GAZETTEER`	0,75	—	Sözlüğün birebir eşleşmesi

İşaretleme ve maskeleme eşiklerinin **ayrı olması zorunlu.** Eşit olsalardı görsel katmanını geçen her yanlış pozitif aynı zamanda maskeleme adayı olurdu. Söylemek geri alınabilir bir yorumdur; kullanıcıya sormadan bulanıklaştırmak onun yerine karar vermektir ve daha yüksek kesinlik ister.

Bu sayılar bağımsız test sonucu değil. Yanlış alarmın paydası 968 değil, kümedeki 512 nötr görsel. Daha önemlisi, 0,90 ve 0,95 eşikleri AYNI 968 görselde taranıp seçildi: küme bu anlamda eşik seçim (geliştirme) kümesidir ve raporlanan oranlar iyimser olabilir. Ayrı tutulmuş bir kümede yeniden ölçüm yol haritasında.

%5,86 hedeflenen %5'in bir tık üstünde ve bu **bilerek** kabul edildi: 0,95'e çıkmak 7 puan duyarlılık götürüp 3 puan yanlış alarm kazandırıyor. Kaçırılan riskli görsel ürünün asıl işini yapmaması demek; fazladan bir afiş ise yalnızca gürültü.

9.3. Ölçülen metrikler ve neden bunlar

- **Kesinlik** — işaretlenen içeriklerin ne kadarı gerçekten ilgili?
- **Duyarlılık** — ilgili içeriklerin ne kadarını yakaladık?
- **Makro F1** — sınıflar dengesiz olduğu için mikro değil makro ortalama.
- **Karışıklık matrisi** — hangi sınıflar birbirine karışıyor?
- **Gecikme** — cihazda karar süresi; akışı takan bir ürün kullanılmaz.
- **Yanlış alarm** — bu üründe en pahalı hata. Kullanıcı bir kez haksız yere uyarıldığında ürünün güvenilirliği biter ve sonraki doğru uyarı da görmezden gelinir.

CLIP ile karşılaştırma — ve neden zayıf bir karşılaştırma

Defterin kendi test bölümünde, **aynı 1.258 görselde** ölçülen tek kayıt:

```
MobileNetV3-Small (bizim, 6,10 MB) : 0,631
CLIP ViT-B/32 sıfır-atış (~350 MB) : 0,574
FARK                               : +0,057
```

Sayı bizim lehimize ama **bu bir başarı göstergesi değil**, çünkü kıyas baştan eğik: CLIP hiç eğitilmedi, bizim model bu dağılımın eğitim bölümünde eğitildi. CLIP'in istemleri elle yazılmış dört İngilizce cümle ve hiç ayarlanmadı. Bu koşullarda gözetimli bir modelin öne geçmesi beklenen sonuçtur.

Zor test kümesinde (968 görsel) CLIP **hiç ölçülmedi**, dolayısıyla oradaki 0,731 ile CLIP arasında bir karşılaştırma yapılamaz.

> **Dürüstlük notu — iddiamız doğruluk değil**
> Ürünün iddiası daha iyi sınıflandırma yapmak değil: **boyut ve gecikme.** Gönderilen dosya 6,10 MB — CLIP'e göre yaklaşık **60 kat küçük** — sunucu gerektirmiyor ve tarayıcıda 17,8 ms'de karar veriyor. Bir tarayıcı eklentisinde çalışabilmenin ön koşulu bu. Raporun ilk sürümü burada "CLIP'i yenmedik" diyor ve iki karşılaştırma değeri veriyordu; o değerlerin hiçbir ölçüm kaydında karşılığı yoktu ve iddia ölçüme göre ters yönde kurulmuştu. Geri çekilen sayılar burada bilerek tekrarlanmıyor — dökümü `docs/olcumler.md` §1u'da.

10. Modelleri tarayıcıda çalıştırmak

Model eğitildikten sonraki iş, eğitimin kendisi kadar kritik ve büyük ölçüde görünmez: Python'da eğitilen bir modelin tarayıcıda **aynı** kararı vermesi kendi başına olmuyor.

10.1. Manifest V3 kısıtları

- İçerik betikleri klasik script olarak yükleniyor; modüller ancak dinamik `import()` ve `web_accessible_resources` ile alınabiliyor.
- WASM için `wasm-unsafe-eval` izni gerekiyor.
- ONNX Runtime Web'in varsayılan dağıtımı çalışma zamanını `blob:` adresinden yüklüyor ve eklenti CSP'si bunu engelliyor; bu yüzden glue kodu gömülü olan `ort.bundle.min.mjs` kullanılıyor.
- Manifest alan sınırları var (açıklama 132 karakter) ve `web_accessible_resources` içinde `**` geçersiz. Bunlar derleme sırasında doğrulanıyor — eklentinin hiç yüklenmemesine yol açan sessiz bir hata olarak yaşandı.

10.2. Ön işleme eşitliği — işin en zor kısmı

Model Python'da (Pillow + torchvision) eğitildi ve orada ölçüldü; çıkarım tarayıcıda yapılıyor. Aradaki tek zorunlu fark **yeniden boyutlandırma**: `canvas.drawImage()` tarayıcının kendi filtresini kullanıyor, Pillow kendi katsayı hesabını ve sabit noktalı tamsayı aritmetiğini. İkisi aynı görselden farklı piksel üretiyor ve fark **sessiz** oluyor — model eğitimde gördüğünden başka bir dağılım görüyor, hiçbir hata mesajı çıkmıyor.

Bu yüzden Pillow'un bilineer yeniden boyutlandırması JavaScript'te birebir yeniden yazıldı: aynı katsayı hesabı, aynı 22 bitlik sabit nokta, aynı iki geçişli sıra, Pillow'un geçiş atlama davranışı dahil. `canvas` yalnızca ham piksel okumak için kullanılıyor, hiç ölçeklemiyor.

Sonuç: 77 kontrol, 8 farklı boyut oranında tensör baytları birebir aynı. Test edilen durumlar arasında yuvarlama sınırı (427,52 → 428), eksen karışması, büyütme (farklı çekirdek), 16:9 küçük resim ve düz renk (katsayı toplamı 1 mi) var.

Bu ölçümün yakaladığı hata

İlk çalıştırmada yeniden boyutlandırma sekiz durumda da birebir tuttu ama **tensör tutmadı**: $1{,}19\times10^{-7}$ fark, yani float32'nin 1 ULP'si. Sebep normalizasyondaydı — NumPy tarafında ortalama bir `float32` dizisi, yani çıkarılan sayı 0,485 değil onun float32 komşusu. JavaScript her sayıyı çift duyarlıkta tutuyor ve `0.485` yazmak **farklı bir operandla** çalışmak demekti. `Math.fround` ile sabitler float32'ye indirildi; fark sıfırlandı.

Ürün açısından 10^{-7} zararsızdı. Ama "birebir aynı" iddiası ya doğrudur ya değildir; arada bir yer yok — ve bu testin varlık sebebi gelecekteki daha büyük bir sapmayı yakalamak.

10.3. Hız — 58 ms'den 27 ms'ye

`Math.fround` her pikselde çağrıldığında ön işleme 1080×1080 bir görselde 58 ms sürüyordu. Girdi bir bayt olduğu için yalnızca 256 olası değer var: kanal başına 256 girişli bir arama tablosu önceden hesaplanıyor ve sonuç **bit düzeyinde aynı** kalırken süre 27 ms'ye iniyor.

10.4. Tarayıcı doğrulaması

Ölçü	Sonuç
Çalışma zamanı + model hazır olma	848 ms
Ortalama çıkarım	**17,8 ms** (v1 modeliyle ölçüldü)
En büyük olasılık farkı (Python ↔ tarayıcı)	**$1{,}27\times10^{-6}$**
Etiketi değişen görsel	**0 / 48**

Ölçü	Sonuç
Eşikte kararı değişen	**0 / 48**

JPEG çözme farkının (tarayıcı ↔ Pillow) ölçülen etkisi ihmal edilebilir çıktı. Python'da ölçülen çalışma noktası tarayıcıda da geçerli.

Sürüm notu: 17,8 ms **v1** modeliyle ölçüldü. Gönderilen **v2** aynı mimari ve aynı giriş boyutu; çıktı eşitliği doğrulandı (fark $1{,}27\times10^{-6}$, 0 karar değişimi) ama gecikmesi yeniden ölçülmedi. Pratikte aynı olması bekleniyor; "ölçüldü" demek yine de tam doğru olmadığı için not düşülüyor.

10.5. Kirlenmiş tuval

Sayfadaki bir `<img> crossorigin` niteliği olmadan yüklendiğinde, CDN doğru başlığı gönderse bile tuval kirleniyor ve `getImageData` `SecurityError` fırlatıyor. Final sürümde iki kademe var: doğrudan okuma → aynı adresi çerezsiz `crossOrigin` ile yeniden isteme. Tasarımda üçüncü bir kademe (servis çalışanının görseli `data:` adresi olarak çözmesi) vardı; ek CDN izni gerektirdiği için finalde kaldırıldı (G05). Hangi kademenin kullanıldığı sayaçla taşınıyor; görsel katmanı çalışmıyorsa sebebinin görünmesi gerekiyor.

Videoda yedek yol yok ve olamaz: yedek yol "aynı adresi yeniden iste" mantığına dayanıyor, videonun adresi ise MediaSource'a bağlı bir `blob:`. Bu durumda erken ve görünür biçimde çıkılıyor — sessizce susmak, ürünün amiral gemisi bileşeni ölüyken her şeyin normal görünmesi demekti.

11. Mahremiyet ve güvenlik

Her şey cihaz üstünde. Sunucu yok, hesap yok, veri dışarı çıkmıyor.

Saklanan	Saklanmayan
Risk olayı metadatası (zaman, tür, skor, güven, kanıt **etiketi**, müdahale, kullanıcı eylemi)	Gönderi ve mesaj metinleri
Oturum sayaçları (süre, tekrar açma, gece kullanımı)	Şifre, kimlik bilgisi, hesap içeriği
Kullanıcı ayarları ve hedefleri	Ekran görüntüsü — hiçbir koşulda alınmıyor
Yanlış tespit geri bildirimi	Kamera, mikrofon, kesin konum

11.1. Ölçüm — gönderi metni gerçekten yazılmıyor mu?

Bu bir iddia ve iddialar ölçülür. Gönderi başlıklarına gerçek kelime olmayan eşsiz işaretler konuldu (`zqxvorbit1`, `plundraq2`, …). Depolamaya yazılan ve arka plana gönderilen **her şey** biriktirilip tek bir dizeye çevrildi ve işaretler arandı. Bir tanesi bile geçse test başarısız sayılıyor.

Ölçü	Sonuç
Analiz edilen gönderi	5
Depolamaya yazılan anahtar	`miraBehavior` — yalnızca oturum sayaçları
Sızan eşsiz işaret	**0**
Sızan başlık parçası	**0**
Toplam incelenen döküm	3.172 karakter
Sayfa dışına giden istek (uçtan uca akış)	**0**

Gönderi metni analiz edilip **sınıf etiketine çevriliyor ve atılıyor.** 60 karakterden uzun tek alan `evidence[0]` ve içeriği bir sınıf etiketi. Karara katkı yapan kelimeler (“Neden gösterildi?” ekranı için) yalnızca bellekte, o kararın ömrü boyunca yaşıyor; depolamaya ve arka plana gitmiyor.

11.2. Tek bilinçli istisna

Popup’taki **“Ayrıntılı günlük”** seçeneği açıldığında gönderi başlığı yerel tarayıcı konsoluna yazılıyor. Varsayılan olarak kapalı, ne yaptığı popup’ta açıkça yazıyor, konsol kalıcı değil ve veri cihazdan çıkmıyor. Olmadan “neden uyarı çıkmadı” sorusu cevaplanamıyor — adaptörün başlığı okuyup okumadığını görmenin başka yolu yok.

11.3. İzin yaklaşımı

Manifest izinleri minimumda: yalnız `storage` ve iki host izni (Instagram, YouTube). `activeTab`, `scripting` ya da tüm sitelere geniş izin istenmiyor; kontrollü demo sayfaları için içerik betiği ayrıca yalnız yerel `/test-pages/` yolunda çalışıyor.

12. Test ve doğrulama

12.1. Tarayıcısız testler

Paket	Sayı	Neyi doğruluyor
Çekirdek	**499**	Risk formülü, eşikler, cooldown, komut katmanı, adaptörler, davranış takibi, örnekleme kuralları
Metin eşitliği	**17**	Python ↔ JavaScript sınıflandırma farkı (en büyük fark 5×10^{-11})
Görsel eşitliği	**77**	Pillow ↔ JavaScript ön işleme, tensör baytları birebir

```
cd core && npm test        # 499 + 17 + 77, tarayıcı gerekmez
```

12.2. Tarayıcı doğrulama sayfaları

Sayfa	Ne kanıtlıyor
`uctan-uca.html`	Gerçek fotoğraftan BLUR komutuna kadar tüm boru hattı
`gorsel-dogrulama.html`	Görsel modelin Python ↔ tarayıcı eşitliği
`mahremiyet.html`	Gönderi metni depolamaya sızıyor mu
`maskeleme.html`	Maskeleme React yeniden çizimine dayanıyor mu (32 kontrol)
`shorts.html`	Paylaşılan oynatıcıda müdahale yaşam döngüsü (17 kontrol)
`instagram.html` · `youtube.html`	Gerçek DOM kopyalarında adaptör doğruluğu

12.3. Uçtan uca sonuç

Ölçü	Sonuç
Nötr fotoğraf, kendiliğinden bulanan	**0 / 12**
Riskli fotoğraf, kendiliğinden bulanan	**12 / 36**
Eşiğin altında geçen görsel	**0**
Sayfa dışına giden istek	**0** (tam çevrimdışı)

Ölçü	Sonuç
Öğe başına analiz (48 öğelik fotoğraf akışı, boşta makine)	19,9 ms (tekrarlarda 20–34 ms)

Süre neye bağlı: 19,9 ms fotoğraf akışında, boşta makinede ölçüldü. Aynı boru hattı yük altında ve maskeleme açıkken öğe başına **299,6 ms**'ye kadar çıkıyor. Video yolu ise çok daha ucuz, çünkü kare 256×379'a indiriliyor: örnek başına 4,8 ms okuma + 7,9 ms ön işleme. Bir gecikme sayısı, ölçüldüğü yük yazılmadan anlamlı değil.

Gece senaryosu ayrıca ölçüldü (saat 02.00, 50 dakika kesintisiz kullanım, kişisel ortanca 13 dakika): en yüksek skor 81, bileşenler arayüz 0,80 · içerik 0,82 · davranış 0,80. Aynı içerik, farklı kullanım — davranış bileşeninin ürüne kattığı fark burada görünüyor. **Bu senaryoda da nötr fotoğraf bulanmadı.**

12.4. Derleme kapıları

- Manifest doğrulama: alan uzunlukları, sürüm biçimi, geçersiz joker karakter, referans verilen dosyaların varlığı.
- Model ihraç kapısı: **ihraç edilen ONNX dosyasının kendisi ölçülüyor**, PyTorch modeli değil. fp32'den 0,02'den fazla sapan bir nicemleme otomatik reddediliyor.
- Veri kapısı: eğitim defterinde eklenen görsel sayısı doğrulanıyor; beklenenin altındaysa hücre duruyor. Yanlış veriyle 30 dakika eğitmek, hiç eğitmemekten pahalı.
- Sayı kapısı: `node tools/denetle.mjs` belgelerdeki her ölçüm sayısı için kayıt arıyor, bulamazsa sıfırdan farklı çıkıyor. Kaynak yalnızca ölçüm kayıtları ve kaynak kodu; README ve bu rapor **denetlenen** taraf.

12.5. Gerçek Instagram akışında doğrulama

Uzantı gerçek `instagram.com` ana akışında çalıştırıldı. Tek oturumda **124 içerik analiz edildi**; sonsuz kaydırma, uzun oturum ve otomatik oynatma dedektörleri tetiklendi, riskli içerik tespit edilip filtrelendi.

Ölçüm ana akışa aittir. Gerçek sonsuz akışta toplam öğe sayısı bilinmediği için bu bir kapsama oranı değil, çalışma doğrulamasıdır.

13. Ölçümün yakaladığı hatalar

Aşağıdaki hataların ortak yanı: çökme yok, hata mesajı yok, ürün çalışıyor görünüyordu. Ölçülmeseler hepsi ürüne girerdi.

- **int8 nicemleme modeli çökertmişti.** Tarayıcıya gönderilecek ONNX her görsele "risk yok" diyordu; makro F1 0,6310'dan 0,1658'e düşüyordu. Metrikler PyTorch modelinden ölçüldüğü için görünmüyordu. İhraç adımı artık gönderilen dosyanın kendisini ölçüyor ve sapma varsa duruyor.
- **Zor negatifler yanlış alarmı iki katına çıkardı.** Nötr örnekler rastgele seçilince manzara ve eşya çıkıyordu; gerçek akış insan fotoğrafıyla dolu. İnhaler, telefon, su içme gibi örnekler eklenince yanlış alarm %4,5'ten %9,4'e çıktı. Ölçüm modeli kötüleştirmedi, gerçek başarımı gösterdi.
- **Kaynak bazlı ayrım modelin ezberlediğini gösterdi.** Satır bazlı bölmede eğitim makro F1 1,0000, test 0,24 çıktı. Kaynak bazlı ayrım modeli görülmemiş şablonlarla sınayınca gerçek başarım ortaya çıktı.
- **Shorts'ta altı videodan beşi analiz edilmiyordu.** YouTube DOM'da üç reel tutuyor ve tek bir oynatıcı paylaşıyor; adaptör bunu bilmediği için yalnızca ilk video işleniyordu.
- **Ürün kendi uyarısını tespit ediyordu.** MİRA'nın uyarı metni de sayfa metnine dahil olduğu için dedektörler onu manipülatif tasarım sanabiliyordu. `data-mira-ignore` niteliği bu yüzden eklendi.

Ortak ders: bu hataların hiçbiri kendi kopya sayfalarımızda görünmemişti, çünkü kopyaları gerçek DOM'a bakarak değil tahmin ederek yazmıştık. Test altyapısının gerçeği ne kadar modellediği, test sayısından daha önemli.

14. Bilinen sınırlar ve yol haritası

Bu bölüm bilerek yazıldı. Jürinin soracağı şeyleri önden söylemek, raporun geri kalanını daha inandırıcı yapar.

14.1. Henüz ölçülmemiş olan

- **Görsel modelinin gerçek video karesindeki İSABETİ.** Boru hattının çalıştığı ölçüldü — kare 256×379'a indiriliyor, örnek başına 4,8 + 7,9 ms, ana iş parçacığının ~%2'si. Ölçülmeyen şey modelin o karelerde ne kadar DOĞRU olduğu: bütün doğruluk sayıları fotoğraf üzerinden ve video karesi farklı bir alan (düşük çözünürlük, hareket bulanıklığı, sıkıştırma). Uzantıya canlı gösterge eklendi, okuma bekliyor. Boru hattının çalışması ile modelin isabetli olması iki ayrı iddiadır ve karıştırılmamalıdır.
- **Gerçek kullanıcıyla pilot test.** Müdahale kabul oranı ve bilinçli çıkış oranı henüz gerçek kullanıcıdan ölçülmedi.
- **Gerçek akışta kapsama.** Instagram'da 124 içerik analiz edildi (§12.5) ama kaç öğenin KAÇIRILDIĞI ölçülmedi. DOM kopyasında payda bizim koyduğumuz için "11/11" denebiliyor; gerçek sonsuz akışta payda yok. Bu bir saha gözlemi, kapsama ölçümü değil.
- **Android deneysel prototipi (0.2.0-experimental).** Rapor sonrası geliştirildi. Emülatör kanıtı var; fiziksel telefon testi yok. Mevcut puanlama yolunda HIGH (≥65) kademesine erişemiyor; mevcut özellik setinin ham tavanı 0,605, yani 61 puan (HIGH eşiği 65). Bu nedenle "HIGH doğrulandı" denemez ve Chrome model metrikleri Android'e taşınamaz.

14.2. Bilinen zayıflıklar

Sınır	Ölçülen değer	Sebep
İngilizce metin	Makro F1 0,548	Eğitimin yalnızca %10'u İngilizce; karakter n-gramları Türkçeye ayarlı

Sınır	Ölçülen değer	Sebep
TOBACCO sınıfı	F1 0,510	Open Images'ta insan doğrulamalı tütün etiketi yalnızca 231 eğitim örneği veriyor
Görsel yanlış alarm	%5,86 @ 0,90	Hedeflenen %5'in bir tık üstünde; duyarlılık lehine bilerek kabul edildi
Koruyucu bağlam sözcüğü	10 metinde ölçüldü (§ olcumler, 24 Eylül)	Önleme/haber sözcüğü (Yeşilay, tedavi, polis, operasyon…) geçen metinde metin ve görsel katmanı çalışmıyor; "Yeşilay destekli deneme bonusu" bu yüzden kaçıyor. Ölçüm: ne metin modeli (saldırılar 0,30–0,37, korunacak metinler 0,07–0,40) ne de bir anahtar kelime kuralı ayırabiliyor — Yeşilay'ın kendi uyarı paylaşımı aynı ifadeleri alıntılıyor. Önleme içeriğini korumak seçildi; çözüm bağlamı ayıran bir model.

14.3. Yol haritası

- Video karesi ölçümü ve gerekirse video verisiyle yeniden eğitim.
- İngilizce metin performansının iyileştirilmesi (veri dengesi).
- TOBACCO sınıfı için ek eğitim verisi.
- Chrome Web Store yayını: Windows kurulum sihirbazı dosyaları yerleştiriyor, ama Chrome'un geliştirici modu ve "paketlenmemiş öğe yükle" adımları hâlâ elle; mağaza paketi bu iki tıkı kaldırır.
- Android deneysel prototipinin fiziksel telefonda sınanması ve HIGH kademesine erişemeyen puanlama yolunun ayrıca ele alınması.

Kaynaklar ve veri lisansları

Kaynak	Kullanım	Lisans
Open Images V7	Görsel modeli — eğitim ve doğrulama, insan doğrulamalı etiketler	CC BY 4.0
Mendeley Smoker Detection	Görsel modeli — **yalnızca değerlendirme**: zor negatifler ve tütün test kümesi	CC BY 4.0
Princeton "Dark Patterns at Scale"	Metin modeli — gerçek, insan etiketli FOMO örnekleri	akademik
`Overfit-GM/turkish-toxic-language`	Metin modeli — gerçek Türkçe nötr örnek	açık
`winvoker/turkish-sentiment-analysis`	Metin modeli — gerçek Türkçe nötr örnek	açık
`onnxruntime-web` 1.19.2	Tarayıcıda ONNX çıkarımı (WASM)	MIT

Ham veri depoya girmiyor; `.gitignore` ile dışarıda tutuluyor.

> **Bu belgedeki sayıların kaynağı**
> Her ölçümün tarihi, yöntemi ve tekrar üretme komutu `docs/olcumler.md` içinde bulunuyor; her bölüm "ne ölçüldü, nasıl ölçüldü, ne çıktı" biçiminde. Bu rapordaki bir sayıyı doğrulamak isteyen oraya bakmalı.