Lojistik Regresyon Yöntemi ile İkili Sınıflandırma

Tuğcan Topaloğlu
Fen Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi
Ankara, Türkiye
topaloglutugcan@gmail.com

***Özet*—Bu rapor, Lojistik Regresyon algoritmasını kullanarak bir sınıflandırma projesi gerçekleştirmektedir. Modelin performansını artırmak ve aşırı öğrenmeyi önlemek amacıyla L2 düzenleme, erken durdurma ve eşik optimizasyonu gibi çeşitli teknikler uygulanmıştır. Sonuçlar, modelin optimal bir F1 skoru elde etme yeteneğini göstermektedir.**

Anahtar Kelimeler — Lojistik Regresyon, İkili Sınıflandırma, L2 Düzenleme, Erken Durdurma, Eşik Optimizasyonu, F1 Skoru, Doğruluk, Kesinlik, Geri Çağırma, Aşırı Öğrenme

I. GİRİŞ

Sınıflama ve regresyon varolan verileri sınıflandırıp sisteme aktarılan yeni verilerin hangi sınıfa ait olduğunu tahminleyebilecek modelleri oluşturan analiz yöntemleridir[1]. Bu metodlardan önemli bir tanesi de lojistik regresyondur. lojistik regresyon her ne kadar regresyon adını alsa da bir sınıflandırma yöntemidir. Bu projede de lojistik regresyon tekniği kullanılmıştır. Aktifleştirme fonksiyonu olarak sigmoid fonkisyonu oluşturulmuş ve çapraz entropi yöntemi ile sınıflandırma yönteminin öğrenmesi gerçekleştirilmiştir. Ardından bu sonuçlar da birden fazla yöntemle daha iyi F1 skoruna sahip olacak şekilde düzenlenmiştir. Yöntemlerin kullanılması her daim modelin F1 skorunun iyileşeceğini göstermese de iyileştirme yöntemleri birlikte veya ayrı kullanımı birbirinden farklı sonuçlar yaratmaktadır.

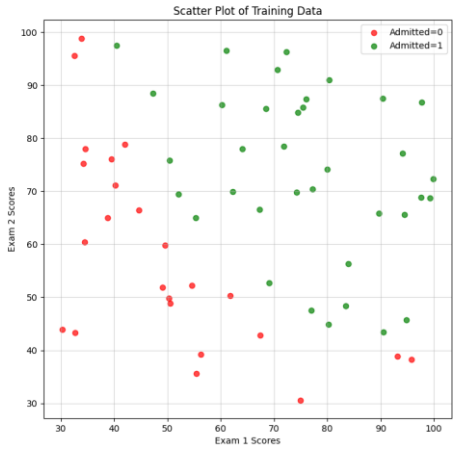
II. DENEYSEL ANALİZ

A. *Verilerin Ayrıştırılması ve İşlenmesi*

Bu projede kullanılan veri seti iki işe alım sınavına giren adayların aldığı puanlar ve işe kabul edilip edilmediklerine dair bilgi içermektedir. Modelin öğrenmesini gerçekleştirmek için öncelikle verilen veri seti %60 eğitim verisi (train_data.txt), %20 doğrulama verisi ve %20 test verisi (test_data.txt) olarak ayrıştırılmıştır. Bu ayrıştırma yöntemi yapılırken seçilen veriler ilgili veri setinden rastgele değerler olarak seçilmiştir. Seçilen bu değerler veri setinden çıkartılıp diğer ayrıştırma işlemi gerçekleştirilmiştir. Bu şekilde toplamda 100 satırlık bulunan veri 60 – 20 – 20 şeklinde bölünmüştür. İşlemler yapılırken modülerliğin sağlanması amacıyla DataHandler adında verileri işleyecek olan bir sınıf oluşturulmuştur. Bu sınıf daha sonra Main sınıfı üzerinde çağrılmıştır. DataHandler sınıfı constructor metodu dışında verilerin ana dosyadan okunduğu DataScrapper metodu, verilerin matplotlib.pyplot kütüphanesi ile görselleştirildiği DataVisualazation metodu ve verileri bölen DataSplitter metodundan oluşmaktadır. DataSplitter metodu verileri sckit-learn kütüphanesi ile bölmektedir. Ayrıca bu metod böldüğü verileri ilgili klasöre yazmaktadır. Bu şekilde veriler bir kez oluşturulup buradan okunabilir. Ancak farklı veriler ile test yapılması istenmesi durumunda ilgili klasördeki metin dosyaları silinerek hiçbir ekstra işlem yapmadan program tekrar çalıştırıldığında yeni veriler oluşturacaktır. DataVisualazation metodu ise oluşturulan bu verilerin iki boyutlu düzlemde nasıl görüldüğünü bizlere göstermektedir bu çıktı şekil 1'de görülebilmektedir.

Şekil. 1. Veri Setinin Görselleştirilmesi

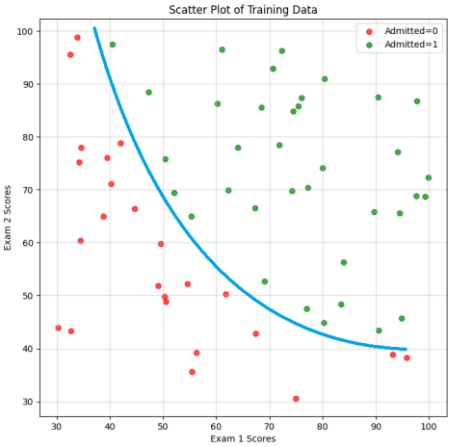
Şekil 1'de görülen yatay eksen ilk sınavı dikey eksen ise ikinci



sınavı göstermekle birlikte kırmızı noktalar kabul edilmeyen adayları yeşil noktalar ise kabul edilen adayları göstermektedir. Modelimizin bu iki sınıfı ayırıp yeni gelen adayların işe alınıp alınmayacağını test etmesi beklenmektedir.

B. *Veri Analizi*

Model eğitimine başlamadan önce görselleştirilen verilerden nasıl bir sonuç beklendiği tahminlenebilir. Burada bu iki sınıfı doğru bir şekilde ayıracak ağırlıklara sahip bir model elde etmeyi amaçlanmaktadır. Bu amaca yönelik model bize şekil 2'de gözüken mavi çizgiye benzer bir sonuç vermesini beklenmektedir. Ancak bu sonuç bu kadar mükemmel olmayabilir. Ayrıca elde edilen sonucun aşırı uyum sağlamadığına da dikkat edilmelidir.



Şekil. 2.Model Eğrisi Tahminlemesi

C. *Modelin Oluşturulması ve Skorların Elde Edilmesi*

Öncelikle modeli eğitmek amacı ile bir aktivasyon fonksiyonu belirlenmesi gerekmektedir. Proje isterlerinde bu fonksiyon sigmoid fonksiyonu olarak belirlenmiştir. Sigmoid fonksiyon denklemi denklem (1)'de görülebilir. Ayrıca ağırlık denklemlerini hesaplamak amacıyla bir loss fonksiyonu kullanılması gerekmektedir. Bu fonksiyon da denklem (2)'de görülebilecek şekilde belirtilen cross entropy loss'tur.

$$f(x) = 1 / (1 + e^{-x}) \quad (1)$$

$$-[y_{target} * \log(y_{predicted}) + (1 - y_{target}) * \log(1 - y_{predicted})] \quad (2)$$

Sigmoid fonksiyonu ile aktive edilen model loss değerlerini denklem (2)'de belirtilen ve cross entropy loss adını alan yöntem ile elde ettikten sonra bu değerleri stochastic gradient descent yöntemi ile $y_{predicted}$ çıkış ağırlıklarını güncellemiştir. Bu yöntemler modülerliğin sağlanması amacıyla LogisticRegression sınıfında hiçbir hazır kütüphane kullanmadan oluşturulmuştur.

1) *Sigmoid Fonksiyonu*

LogisticRegression sınıfında bulunan ve standart euler sayısı belirlenen fonksiyon denklem (1)'de görüldüğü şekilde oluşturulmuştur.

2) *Cross Entropy Loss Fonksiyonu*

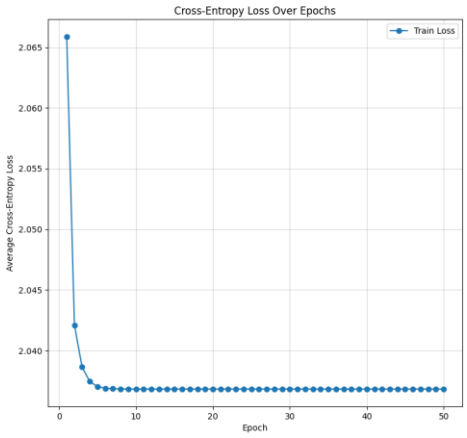
Fonksiyon denklem (2)'de görüldüğü şekliyle LogisticRegression sınıfında oluşturulmuştur. Fonksiyona alınan değerler belirlenen epsilon değeri yardımıyla 0 ile 1 aralığına normalize edilmiştir.

3) *Stochastic Gradient Descent*

StochasticGradientDescent fonksiyonu, Lojistik Regresyon modeli için Stokastik Gradyan İnişi (SGD) algoritmasını uygular. Giriş özellikleri, etiketler, ağırlıklar, öğrenme oranı ve epoch sayısını alarak her epoch boyunca ağırlıkları günceller ve kayıp değerlerini hesaplar. Her veri örneği için lineer kombinasyon hesaplanır, sigmoid fonksiyonu ile tahmin yapılır ve çapraz entropi kaybı kullanılarak kayıp değeri bulunur. Gradyanlar hesaplanarak ağırlıklar güncellenir ve her epoch'un ortalama kaybı kaydedilir. Eğitim sonunda ağırlıklar ve epoch kayıpları dosyaya yazılır. Bu süreç, modelin performansını optimize etmek ve öğrenme eğrisini analiz etmek için kullanılır. LogisticRegression sınıfı altında bulunan bu fonksiyon eğitilen tüm modellerde kullanılabilecek şekilde modülerdir.

4) *Train Verisi ile Model Eğitimi ve Sonuçları*

Veri setinden ayrıştırılan veri ile main sınıfı içerisinde eğitimler yapılmıştır. Yapılan eğitimlerde bir çok değer denendikten sonra değerler learning_rate = 0.001 ve epoch = 50 olarak belirlenmiştir. Stochastic Gradient Descent yöntemi kullanılarak $y_{predicted}$ ağırlıkları güncellenmiş ve şekil 3'te bulunan grafik elde edilmiştir.



Şekil. 3.Train Modeli Epoch Başına Cross-Entropy Loss

Grafik incelendiğinde loss değeri ilk 10 epoch'ta düşmekte be daha sonrasında az miktarlarda değişmektedir bu yüzden 10. epoch civarında durdurma işlemi yapılabilir ancak bazı değerlerin değiştiği gözlemlendiğinden bu değer geliştirilmiş modelde biraz daha farklı belirlenmiştir. Yani loss değeri başlangıçta hızlı bir şekilde azalıyor. Daha sonra eğri yavaşlıyor ve bir süre sonra sabitleniyor. Değer yaklaşık Average Cross-Entropy Loss = 2.03 olacak şekilde sabitlenmektedir (ilgili süreç ve değerler /data/epoch_loss_output.txt dosyasından izlenebilir ve görülebilir). Bu durumda gerçekleştirdiğimiz bu erken durdurma optimizasyonuna gidilebilir bir sonraki adımlarda bu işlemi de yapacağız. Model sonucunda final ağırlıklar [0.057334261326887945, 0.01055432201850106] şeklinde bulunmuştur (değerler /data/weigts.txt dosyasından

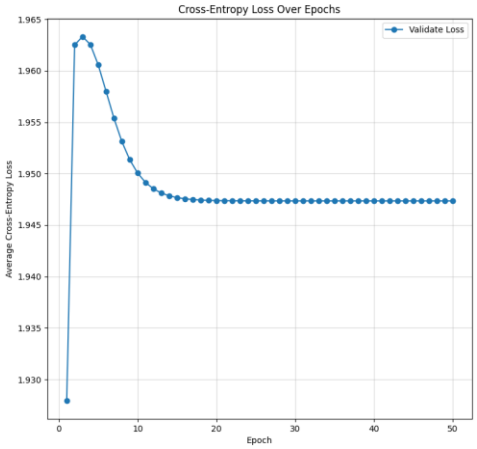
gözlemlenebilir). Tablo 1'deki metrikler incelendiğinde modelin veriyi anladığını ancak bazı optimizasyonlar gerektiğini görüyoruz

TABLO I. TRAİN MODELİ TEST VERİSİ ÜZERİNDEKİ METRİKLERİ

Model Adı	Metrikler				Threshold Değeri
	Accuracy	*Precision*	*Recall*	*F1-Score*	
Train Model	%60.00	%60.00	%100.00	%75.00	Standart = 0.5

5) *Validate Verisi ile Model Eğitimi ve Sonuçları*

Train seti ile aynı şekilde eğitimler yapılmıştır. Yapılan eğitimlerde train setiyle aynı değerler olan learning_rate = 0.001 ve epoch = 50 olarak belirlenmiştir. Stochastic Gradient Descent yöntemi kullanılarak $y_{predicted}$ ağırlıkları güncellenmiş ve şekil 4'te bulunan grafik elde edilmiştir.



Şekil. 4.Validate Modeli Epoch Başına Cross-Entropy Loss

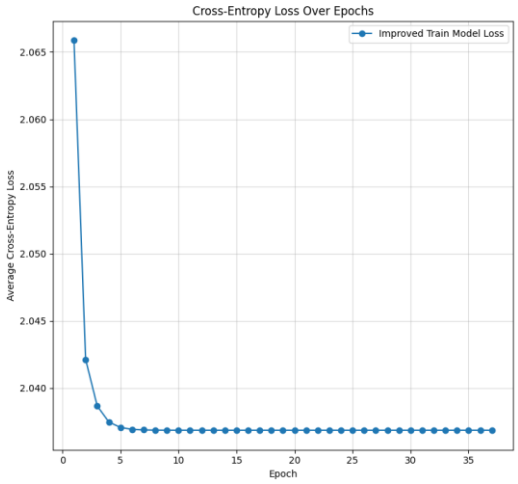
Grafiği incelediğimizde başlangıçta bir dalgalanma görüyoruz doğrulama kaybı başlangıçta bir miktar artış göstermiş (muhtemelen modelin henüz optimal bir dengeye ulaşmaması nedeniyle) ancak yaklaşık epoch 15'ten sonra doğrulama loss değeri stabilize olmuş. Ancak vallidate loss'u train loss'una göre daha yüksek bir seviyede sabitlenmiş. Bu durum modelin eğitim verisine biraz fazla uyum sağladığını (hafif overfitting) gösterebilir. Bu durumda overfitting'i engellemek için geliştirme yapılabilir. Bir sonraki adımda erken durdurma ve L2 düzenlemesi ile birlikte bu overfitting önlenecektir. Bu model için yaklaşık epoch = 5 ve Average Cross Entropy Loss = 1.9580 civarında erken durdurma işlemi yapılacaktır (ilgili süreç ve değerler /data/epoch_loss_output.txt dosyasından izlenebilir ve görülebilir). Model sonucunda final ağırlıklar [0.019140150164792855, 0.08752730398529741] şeklinde bulunmuştur (değerler /data/weigts.txt dosyasından gözlemlenebilir).

TABLO II. VALİDATE MODELİ TEST VERİSİ ÜZERİNDEKİ METRİKLERİ

Model Adı	Metrikler				Threshold Değeri
	Accuracy	*Precision*	*Recall*	*F1-Score*	
Train Model	%60.00	%60.00	%100.00	%75.00	Standart = 0.5

6) *Geliştirilmiş Train Verisi ile Model Eğitimi ve Sonuçları*

Bir önceki train verisi üzerinde hep geliştirmeler hem de projede istenilen erken durdurma işlemi yapılmıştır. Eğitim kaybı ilk birkaç epoch'ta hızla azalmış ve bu beklenen bir davranış. Model eğitim setindeki desenleri hızla öğreniyor. Bu modelde erken durdurma işlemi uygulandığı için maksimum verimi görebilmek adına epoch = 1000 olarak değiştirilmiştir ancak learning_rate = 0.001 olarak sabit bırakılmıştır. İlk işleme benzer bir grafik oluşmuş ve model şekil 5'teki grafikte görüldüğü üzere epoch = 37'de durmuştur. Bu işlemlerin tamamının tüm detayları ilk modelde bahsedilen dosyalarda tekrardan bulunabilir. Bu durdurma güncellemesinin ardından en efektik thresholdu bulmak için CalculateBestThreshold metodu oluşturulmuştur bu metod 0.1'den değerinden 1.0 değerine 0.01 adımla tüm threshold değerlerini deneyerek maksimum F1 skoru alınabilecek threshold değerini buluyor. Ardından bu değer ile metrikleri hesaplıyor.



Şekil. 5.Geliştirilmiş Train Modeli Epoch Başına Cross-Entropy Loss

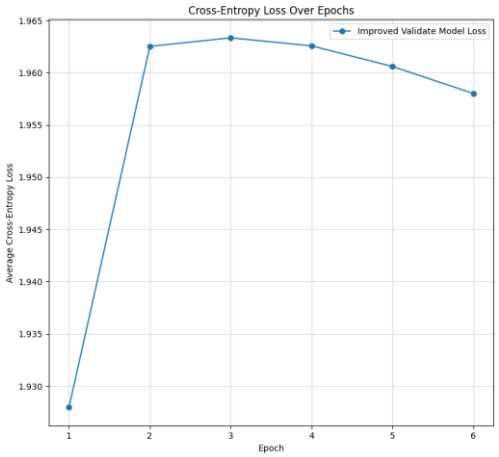
Geliştirme amacıyla ayrıca L2 regularization tekniği kullandık ve şekil 5'teki grafikte görüldüğü üzere stabilizasyon süreci oldukça düzgün bu da L2 regularization'ın etkili olduğunu ve ağırlıkların aşırı büyümesini engellediğini gösteriyor. Threshold bulan metodumuzla birlikte bu işlemleri gerçekleştirdiğimizde tablo 3'te bulunan F1 skorunu ve metriklerini elde ederek başarıyı arttırdık. Ağırlıklar [0.057322637068231444, 0.01055987097242865] şeklinde güncellenmiş oldu (detaylı ağırlıklar /data/weights.txt dosyasında bulunabilir). Average Cross-Entropy Loss yaklaşık 2.03 değerinde devam etmektedir.

TABLO III. GELİŞTİRİLMİŞ TRAİN MODELİ TEST VERİSİ ÜZERİNDEKİ METRİKLERİ

Model Adı	Metrikler				Yaklaşık Threshold Değeri
	Accuracy	*Precision*	*Recall*	*F1-Score*	
Improved Train Model	%85.00	%80.00	%100.00	%88.89	0.97

7) Geliştirilmiş Validate Verisi ile Model Eğitimi ve Sonuçları

Şekil 6 incelendiğinde başlangıçta doğrulama loss'unda bir artış görülüyor (epoch 1-3 civarı) ardından epoch 5-6'dan itibaren düşüşe geçiyor. Bu modelin başlangıçta doğrulama setinde kötü performans gösterdiğini ancak sonrasında regularization ve eğitim sürecinin doğrulama setine uyum sağlamasına yardımcı olduğunu gösteriyor.



Şekil. 6.Geliştirilmiş Validate Modeli Epoch Başına Cross-Entropy Loss

Eğitim ve doğrulama loss grafikleri arasındaki fark oldukça azalmış. Bu, L2 regularization ve early stopping sayesinde modelin aşırı öğrenmesinin önlendiğini işaret ediyor. Bu iyileştirmeler değerleri daha önceki metriklerin üzerine çıkararak tablo 4'te görünen metriklere yükseltiyor. Bu da yaptığımız iyileştirmenin başarılı bir sonuç verdiğini gösteriyor. Model ağırlıkları [0.023708153086232815, 0.08467290376836599] şeklinde güncellenmiş oldu. Average Epoch Loss ise hesaplanan 1.9580 değerine gelmiş oldu.

TABLO IV. GELİŞTİRİLMİŞ VALİDATE MODELİ TEST VERİSİ ÜZERİNDEKİ METRİKLERİ

Model Adı	**Metrikler**				**Yaklaşık Threshold Değeri**
	Accuracy	***Precision***	***Recall***	***F1-Score***	
Improved Train Model	%65.00	%63.16	%100.00	%77.42	0.98

III. SONUÇ

Bu çalışmada, Lojistik Regresyon algoritmasını kullanarak bir sınıflandırma modeli geliştirilmiş ve çeşitli teknikler uygulanmıştır. Modelin performansı doğruluk, kesinlik, geri çağırma ve F1 skoru gibi metriklerle değerlendirilmiş ve kullanılan L2 düzenleme erken durdurma ve eşik optimizasyonu gibi tekniklerin etkili olduğu gösterilmiştir. Eğitim aşamasında stochastic gradient descent yöntemiyle model ağırlıkları optimize edilmiş ve F1 skoru başlangıçta %75 seviyesindeyken, geliştirilmiş modelde %88.89'a yükselmiştir. L2 düzenleme sayesinde aşırı öğrenmenin önüne geçilmiş eğitim ve doğrulama loss grafikleri arasındaki farkın azalması modelin daha iyi genelleme yapabildiğini ortaya koymuştur. Doğrulama ve test veri setleri üzerindeki analizler, modelin farklı veri setlerine uyum sağladığını göstermiş ve doğrulama setinde elde edilen %77.42'lik F1 skoru yapılan iyileştirmelerin etkinliğini kanıtlamıştır. Sonuç olarak, bu çalışma, Lojistik Regresyon algoritmasının temel sınıflandırma problemleri için etkili bir yöntem olduğunu ve doğru optimizasyon teknikleriyle performansının önemli ölçüde artırılabileceğini ortaya koymuştur. Gelecekte modelin daha büyük veri setleri üzerinde test edilmesi ve veri sınıf dengesizliklerini gidermek için ek yöntemlerin uygulanması, performansın daha da geliştirilmesine olanak sağlayabilir..

KAYNAKLAR

[1] Özekes, S., & Çamurcu, Y. (2002). Veri madenciliğinde sınıflama ve kestirim uygulaması. Marmara Üniversitesi Fen Bilimleri Dergisi, 18, 1-17.