Türkiye Klinikleri Biyoistatistik Dergisi
Turkiye Klinikleri Journal of Biostatistics Turkiye Klinikleri J Biostat. 2024;16(2):84-94

ORİJİNAL ARAŞTIRMA ORIGINAL RESEARCH DOI: 10.5336/biostatic.2024-101686

Alternatif Uç Birleştirme Bölgelerinin Makine Öğrenimi ve Derin Öğrenme Yöntemleriyle Tahmin Edilmesinde Örnek Genişliğinin Etkisi: Metodolojik Bir Çalışma

Effect of Sample Size on Predicting Alternative Splicing Regions Using Machine Learning and Deep Learning Methods: A Methodological Study

Ragıp Onur ÖZTORNACI[a]

[a]Koç Üniversitesi Translasyonel Tıp Araştırma Merkezi, Biyoistatistik AD, İstanbul, Türkiye

Bu çalışma, 24. Ulusal ve 7. Uluslararası Biyoistatistik Kongresi'nde (26-29 Ekim 2023, Afyonkarahisar) sözlü olarak sunulmuştur.

ÖZET Amaç: Alternatif uç birleştirme, genlerin RNA işlenmesi sırasında farklı kombinasyonlarda birleştirilmesi sürecidir. Bu süreç, bir genin kodladığı proteinin farklı formlarının oluşturulmasını sağlar. Bu çalışmanın amacı, alternatif uç birleştirme bölgelerinin tespiti için hangi modelin daha yüksek doğrulukla sonuç verdiğini tespit etmektir. **Gereç ve Yöntemler:** Simülasyonlar Python programlama dili ile gerçekleştirilmiş olup örnek büyüklükleri 25, 50, 100, 150 ve 200 olarak belirlenmiştir. Genetik veri setlerinde A, C, G ve T nükleotitlerinin frekansları eşit olarak dağıtılmıştır. Ayrıca uç birleştirme bölgeleri GTAGC, GTAGT, GTAGA ve GTCGA olarak belirlenmiş ve simülasyon 10.000 kez tekrarlanmıştır. Tüm veri setlerinde, destek vektör makineleri, rastgele orman (RF), long short-term memory (LSTM) ve derin sinir ağları [deep neural networks (DNN)], yöntemleri için aşırı öğrenme (overfitting) problemi önlenmesi amacıyla %67,5 eğitim veri seti, %10 test veri seti ve %22,5 doğrulama seti olarak belirlenmiş ve modellerin tahmin gücü test edilmiştir. Grafikler için R programlama dili kullanılmıştır ve tüm işlemler Unix işletim sistemi kullanılarak yapılmıştır. **Bulgular:** Örneklem büyüklüğü arttıkça modellerin performansının da arttığı gözlemlenmiştir. Özellikle DNN ve LSTM modellerinin performansı, örneklem büyüklüğünün artmasıyla birlikte istikrarlı bir şekilde yükselmektedir. LSTM modeli, genellikle diğer modellere göre daha yüksek F1-skor, spesifite ve sensitivite değerlerine sahiptir. RF modeli, genellikle yüksek örneklem büyüklüklerinde etkili bir şekilde çalışmaktadır. **Sonuç:** Bu çalışma, alternatif uç birleştirmelerin belirlenmesi sürecinde, makine öğrenimi ve derin öğrenme modellerinin etkili bir şekilde kullanılabileceğini ortaya koymaktadır.

Anahtar kelimeler: Alternatif uç birleştirme; derin öğrenme; makine öğrenimi

ABSTRACT Objective: Alternative splicing, the process of generating diverse protein isoforms by combining genes in various ways during RNA processing, is a crucial mechanism in cellular function. This study aims to identify the model with the highest accuracy in detecting alternative splicing regions. **Material and Methods:** Simulations were executed in Python with sample sizes ranging from 25 to 200. Genetic datasets maintained even distributions of A, C, G, and T nucleotides. Splicing regions were defined as GTAGC, GTAGT, GTAGA, and GTCGA, with 10,000 repetitions of the simulation. To prevent overfitting, 67.5% of the data served as the training set, 10% as the test set, and 22.5% as the validation set for support vector machines, random forest (RF), long short-term memory (LSTM), and deep neural network (DNN) methods. Model performance was assessed using prediction metrics. R programming language and computations were conducted on a Unix system. **Results:** Increasing the sample size correlated with enhanced model performance, notably for DNN and LSTM models. The LSTM model consistently demonstrated superior F1-score, specificity, and sensitivity compared to other models. The RF model exhibited effective performance, particularly with larger sample sizes. **Conclusion:** This study underscores the effectiveness of machine learning and deep learning models in identifying alternative splicing events. The findings emphasize the importance of considering model choice and sample size in optimizing the accuracy of alternative splicing region detection during RNA processing studies.

Keywords: Alternative splicing; deep learning; machine learning

KAYNAK GÖSTERMEK İÇİN:
Öztornacı RO. Alternatif uç birleştirme bölgelerinin makine öğrenimi ve derin öğrenme yöntemleriyle tahmin edilmesinde örnek genişliğinin etkisi: Metodolojik bir çalışma. Turkiye Klinikleri J Foren Sci Leg Med. 2024;16(2):84-94.

Correspondence: Ragıp Onur ÖZTORNACI
University of Bristol, The MRC Integrative Epidemiology Unit, Department of Population Health Sciences, Bristol, UK
E-mail: onur.oztornaci@gmail.com

Peer review under responsibility of Turkiye Klinikleri Journal of Biostatistics.

Received: 25 Jan 2024 ***Received in revised form:*** 15 Feb 2024 ***Accepted:*** 27 Mar 2024 ***Available online:*** 13 May 2024

Son yıllarda biyoistatistik ve biyoinformatik alanında yapılan çalışmalar, genlerin işlevlerini anlamak ve farklı işlevlere sahip genlerin hücreler üzerindeki etkilerini ortaya çıkartmak üzere kurgulanmaktadır. Bu çalışmalar arasında alternatif uç birleştirme süreci, genlerin RNA işlenmesi sırasında farklı kombinasyonlarda birleştirilmesini sağlayarak bir genin kodladığı proteinin birçok formunun oluşmasını mümkün hâle getirir. Bu mekanizma, hücrelerin çeşitli koşullara ve işlevlere uyum sağlamasında temel rol oynar.[1,2] Alternatif uç birleştirmenin doğru bir şekilde tanımlanması, hücresel işlevlerin ve organizmalardaki biyolojik süreçlerin anlaşılması için kritik öneme sahiptir. Ayrıca alternatif uç birleştirme normal hücresel süreçlerin yanı sıra patojenik olarak hastalıkların meydana gelmesinde önemli bir rol oynamaktadır. Dolayısıyla alternatif uç bölgelerin doğru bir şekilde tespit edilmesinin, kişiselleştirilmiş tıp alanında tedavi sürecine önemli ölçüde katkı sağlayabileceği düşünülmektedir. Son yıllarda, makine öğrenimi (MÖ) ve derin öğrenme (DÖ) gibi veri analizi yöntemleri büyük bir ivme kazanarak yaygın analiz yöntemleri hâline gelmektedir.[3,4] Bu yöntemler, büyük veri setlerini hızla analiz edebilme yetenekleriyle, alternatif uç birleştirmenin tespitinde önemli bir avantaj sağlamaktadır.[4] Bu çalışmanın temel amacı, alternatif uç birleştirme bölgelerinin tespiti için farklı MÖ ve DÖ modellerinin performansını karşılaştırmaktır. Bu amaçla, farklı örnek büyüklüklerinde ve genetik veri setlerinde çeşitli algoritmalar kullanılarak yapılan simülasyonlarla bu modellerin performansı değerlendirilmiştir. Bu çalışmanın sonuçları, alternatif uç birleştirmenin tespiti alanında hangi modelin daha yüksek doğrulukla sonuç verdiğini belirlememiz açısından önemli bir adım olacaktır. Aynı zamanda bu çalışmanın kurgusu gereği MÖ ve DÖ yöntemlerinin performans değerlendirme kriterleri için önemli bilgiler sağlamaktadır ve de genetik araştırmalarda MÖ ve DÖ yöntemlerinin etkili bir şekilde kullanılabileceğini göstererek, bu alandaki gelecekteki çalışmalara da ışık tutacaktır.

GEREÇ VE YÖNTEMLER

Bu çalışma, metodolojik bir çalışma olarak planlanmıştır. Bu çalışmada kullanılan genetik veri setleri, simülasyon ile elde edilmiştir. Yapılan simülasyonlarda üretilen veriler gen dizileri, FASTA formatına uygun olarak tasarlandı. Veri setlerindeki A, C, G ve T nükleotitlerinin frekansları eşit olarak dağıtılmıştır. Alternatif uç birleştirme bölgeleri, GTAGC, GTAGT, GTAGA ve GTCGA olarak belirlendi. Bu bölgeler, literatürde belirtilen standart tanımlara uygun olarak seçildi. Simülasyonlar, Python programlama dili kullanılarak gerçekleştirildi ve grafikler için R programlama dilinin ggplot2 paketi kullanıldı.[5-7] Örnek büyüklükleri (25, 50, 100, 150, 200) olarak belirlendi. Her örnek büyüklüğü için veri seti rastgele olarak, %67,5'i eğitim seti, %10'u test seti ve %22,5'i doğrulama seti olarak ayrıldı ve simülasyon 10.000 kez tekrarlanmıştır. İşlemler Unix işletim sistemi üzerinde yapılmıştır (Tablo 1).

İlgili bölümler; Dizi_Üret (Uzunluk): Belirli bir uzunluktaki genetik diziyi oluşturan bir fonksiyonu tanımlar. Nükleotid_Frekansları = {'A': 0.25, 'C': 0.25, 'G': 0.25, 'T': 0.25} A, C, G ve T nükleotidlerinin frekanslarını belirler. Alterlatif_Uç_Bölgeleri = ("GTAGC", "GTAGT", "GTAGA", "GTCGA") Alternatif uç bölgeleri olarak adlandırılan dizilere ait seçenekleri içeren bir listedir. Alterlatif_Uç_Bölgeleri_Olasılıkları = (0.25, 0.25, 0.25, 0.25) Alternatif uç bölgelerinin seçilme olasılıklarını belirten bir listedir. Dizi = [] Genetik diziyi temsil eden bir boş listedir. DÖNGÜ: Sonsuz döngüyü başlatır (Bu çalışmada 10.000 olarak belirlenmiştir). Nükleotid = rastgele.seçim (liste(Nükleotid_Frekansları.keys()), liste (Nükleotid_Frekansları.values())) [0]: Nükleotid frekanslarına göre rastgele bir nükleotid seçer. Dizi.ekle(Nükleotid) Seçilen nükleotidi dizinin sonuna ekler. EĞER Dizi.uzunluk == Uzunluk: Dizi belirlenen uzunluğa ulaştığında: ÇIKIŞ ". birleştir (Dizi) ifadesiyle diziyi birleştirip çıkış yapar. Genler = [] Gen adlarını depolamak için bir liste oluşturur. DÖNGÜ i 0'dan n'ye KADAR: 0'dan n'ye kadar olan sayılar arasında bir döngü başlatır. Gen_Adı = "Gen" + STR(i) Her bir gen için ad oluşturur. Genler.ekle(Gen_Adı) Oluşturulan gen adını

Genler listesine ekler. Num_Varyantlar = N Oluşturulacak varyant sayısını belirler. Dizi_Uzunluğu = S Oluşturulan genetik dizilerin uzunluğunu belirler. Veri = [] Genetik diziler ve gen adlarını depolamak için bir liste oluşturur. HER Gen_Adı İÇİN Genler: Genler listesindeki her bir gen için: DÖNGÜ _ 0'dan Num_Varyantlar'a KADAR: 0'dan Num_Varyantlar'a kadar olan sayılar arasında bir döngü başlatır. Dizi = Gerçekçi_Dizi_Üret(Dizi_Uzunluğu) Önceki fonksiyonu kullanarak gerçekçi bir genetik dizi oluşturur. Veri.ekle({'Gen': Gen_Adı, 'Dizi': Dizi}) Oluşturulan gen adı ve genetik diziyi Veri listesine ekler (Tablo 1).

TABLO 1: Veri simülasyonu örnek algoritması.

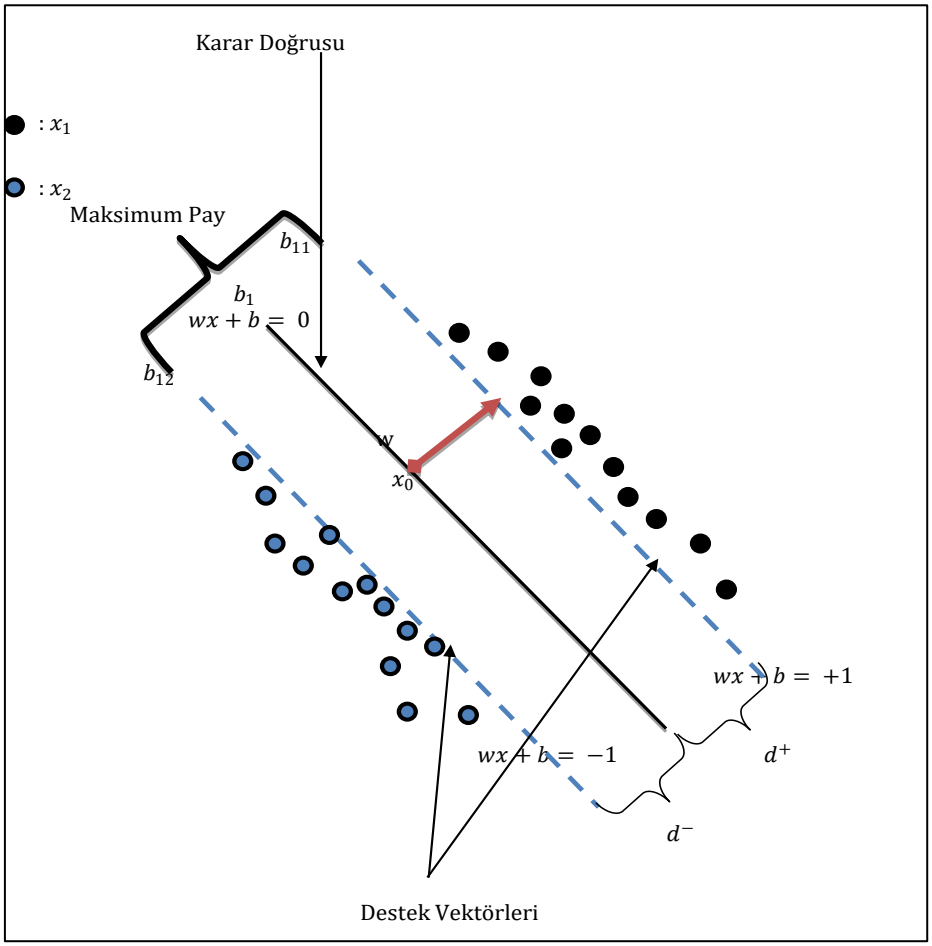
Veri Simülasyonu Örnek Algoritması	
1.	Dizi_Üret (Uzunluk):
2.	Nükleotid_Frekansları = {'A': 0.25, 'C': 0.25, 'G': 0.25, 'T': 0.25}
3.	Alterlatif_Uç_Bölgeleri = ["GTAGC", "GTAGT", "GTAGA", "GTCGA"]
4.	Alterlatif_Uç_Bölgeleri _Olasılıkları = [0.25, 0.25, 0.25, 0.25]
5.	Dizi = []
6.	DÖNGÜ:
7.	Nükleotid = rastgele.seçim(liste(Nükleotid_Frekansları.keys()),
8.	liste(Nükleotid_Frekansları.values()))[0]
9.	Dizi.ekle(Nükleotid)
10.	EĞER Dizi.uzunluk == Uzunluk:
11.	ÇIKIŞ ''. birleştir (Dizi)Genler = []
12.	DÖNGÜ i 0'dan n'ye KADAR:
13.	Gen_Adı = "Gen" + STR(i)
14.	Genler.ekle(Gen_Adı)
15.	Num_Varyantlar = N
16.	Dizi_Uzunluğu = S
17.	Veri = []
18.	HER Gen_Adı İÇİN Genler:
19.	DÖNGÜ _ 0'dan Num_Varyantlar'a KADAR:
20.	Dizi = Gerçekçi_Dizi_Üret(Dizi_Uzunluğu)
21.	**Veri.ekle({'Gen': Gen_Adı, 'Dizi': Dizi})**

Böylelikle genetik veride temel yapıyı oluşturan nükleotidler oluşturulur (A, T, G, C) ve daha sonra bu nükleotidlerin belirli sıralarla bir araya gelmesi ile oluşan alternatif uç bölgeleri oluşturulan nükleotid dizilerine eklenir. Son olarak rastgele gen isimleri oluşturularak bu dizilere eklenir. Veri oluşturma aşaması Python programlama dilini Unix işletim sisteminde Bash script olarak çalıştırılmıştır.

MÖ MODELLERİ

Destek Vektör Makineleri

Vladimir Vapnik ilk olarak 1992 sınıflama problemleri için Lagrange çarpanlarını kullanarak sınıfları arası maksimum ayrıma yeteneğine sahip minimum hata ile sınıflar arasına hayali bir çizgi çekmeyi hedefleyen algoritma geliştirmiştir. Bu durum, Şekil 1 ile ifade edilmiştir.



ŞEKİL 1: Destek vektör makineleri.

$$\mathbf{min}_{\boldsymbol{\omega},\boldsymbol{b},\boldsymbol{\xi}} \frac{\mathbf{1}}{\mathbf{2}} \boldsymbol{\omega}^{\boldsymbol{T}} * \boldsymbol{\omega} + \boldsymbol{C} * \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{\xi}_{\boldsymbol{i}} \quad \textbf{(1)}$$

ω_m, b, ξ, V değişkenleri için,

$$y_i * f_c(x_i) \geq 1 - \xi_i \quad (2)$$

$\forall_i \; ve \; \xi_i \geq 0$ olmak üzere, (1 ve 2) numaralı denklemlerde C regülasyon parametresi, ξ aylak değişken V ise ayar parametresi ve ω ise ağırlıklardır. Denklemde optimizasyon çözümünü elde etmek için kullanılacak olan en küçük toplamları bulmak için Lagrange çarpanlarından yararlanılır ve ağırlık vektörlerinin de yardımıyla sınıflandırma için iki sınıf arasındaki maksimum mesafeyi belirlemek için, $wx_k + b = -1 \; ve \; wx_l + b = +1$ eşitlikleri kullanılır ve bu eşitlikler birbirlerinden çıkartılarak, maksimize edilmek istenir (Şekil 1). Destek vektör makineleri, eğitim süresi olarak diğer algoritmalarla kıyaslayınca uzun sürmesine rağmen destek vektör makineleri aşırı uyum problemine karşı dirençlidir.[8-10]

Rastgele Orman

En temel MÖ olan karar ağaçlarının bir araya getirilerek oluşturduğu hem sürekli yapıdaki değişkeneler için kullanılabilen hem de kategorik yapıdaki değişkenler için kullanılabilen rastgele orman algoritması 2001 yılında Breiman tarafından bulunmuştur. Eksik verilerin tamamlanması, özellik seçimi ve modele katkı sunan değişkenlerin ağırlıklandırılması gibi birçok kullanım alanı vardır.[11]

$$\sum\sum_{i=1}^{n}(f(G_i.T)|T|)(f(G_i,T)|T|) \quad (3)$$

Formülasyonda (3), T eğitim setini temsil ederken, G_i sınıfına ait olarak seçilme olasılığının fonksiyonu $(f(G_i.T)|T|)$'dir. Rastgele orman algoritmasının diğer bir güçlü yanı da modele katkı sunan değişkenlerin "değişken önemlilik (variable importance)" skoru ile hesaplanabilmesidir. Bu skor üç farklı şekilde, out-of-bag örnekleme yöntemi, bootstrap örnekleme yöntemi ve z-skoruna dayanan hesaplama yöntemi ile hesaplanabilir.

$$VI^{(t)}(x_j) = \frac{\sum_{i\in\bar{\mathfrak{B}}(t)} I(y_i=\hat{y}_i^{(t)})}{\bar{\mathfrak{B}}(t)} - \frac{\sum_{i\in\bar{\mathfrak{B}}(t)} I(y_i=\widehat{y_{i,\pi_j}}^{(t)})}{\bar{\mathfrak{B}}(t)} \quad (4)$$

Burada (4), $\bar{\mathfrak{B}}(t)$ ifadesi bir t ağacı için out-of-bag örnekleme yöntemi ile elde edilen örneklem, t ∈ {1, 2, ...,n.ağaç} olmak üzere, $\hat{y}_i^{(t)} = f^{(t)}(X_i)$'dir. Burada, tahmin sınıfındaki i. gözlem tahmin sınıfından yer almadan önce ve $\widehat{y_{i,\pi_j}}^{(t)} = f^{(t)}(X_i,\pi_j)$'dir. $\widehat{y_{i,\pi_j}}^{(t)} = f^{(t)}(X_i,\pi_j)$ tahmin sınıfını olmak üzere i. gözlemin j. gözleme kadar permütasyonu ve $X_i,\pi_j = (X_{i,1},X_{i,2},\ldots\ldots,X_{\pi_j(i),j},X_{\pi_j,j+1},..,X_{i,p})$'dir. $VI^{(t)}$ değişkenin önem skorudur. $VI^{(t)}$'nin sıfıra eşit olması durumunda, eğer X_j değişkeni t ağacında yer almamaktadır. Bootstrap yöntemiyle devam edecek olursak,

$$VI(x_j) = \frac{\sum_{t=1}^{n-ağaç\ sayısı} VI^{(t)}(x_j)}{n-Ağaç\ Sayısı} \quad (5)$$

Burada (5), $VI^{(t)}(x_j)$ skoru birbirinden bağımsız n-ağaç sayısından elde edilir. Son olarak z skoru yöntemi ile hesaplanmak istenirse,

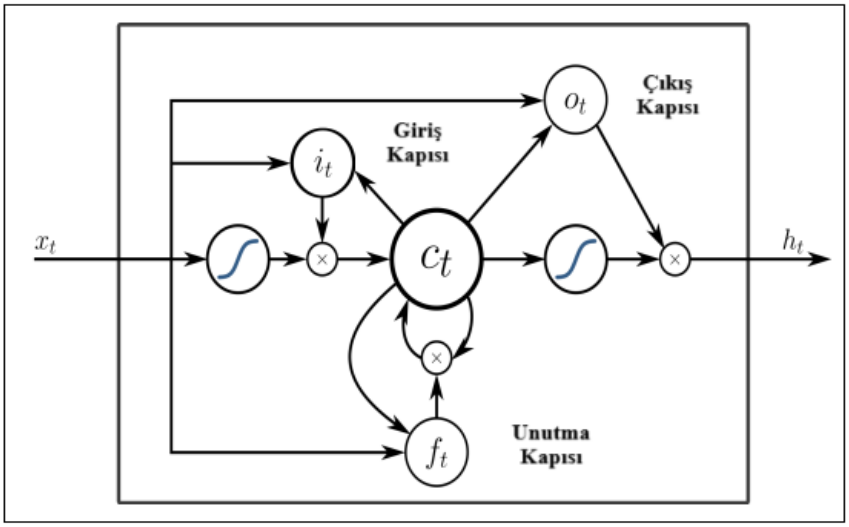
$$\widetilde{VI}(x_j) = \frac{VI(x_j)}{\frac{\hat{\sigma}}{\sqrt{n-Ağaç\ Sayısı}}} \quad (6)$$

Formülasyonu (6) ile hesaplanır.[11,12]

DÖ MODELLERİ

Uzun Kısa Dönem Hafıza Algoritması

Son dönemlerde oldukça popüler hâle gelen Long Short-Term Memory (LSTM), özellikle zaman serisi verileri gibi dizisel veri yapılarında etkili olan bir algoritmadır. LSTM, geleneksel sinir ağlarından farklı olarak, zaman bağımlılığı ve uzun dönemli bağımlılıkları daha etkili bir şekilde ele alabilen bir tür tekrarlayan sinir ağıdır.[13] LSTM temel konsept olarak 3 bölümden oluştuğu düşünülebilir. Bunlar, hücre durumu (c_t), gizli durum (h_t) ve 3 kapı [unutma kapısı (f_t), giriş kapısı (i_t), çıkış kapısı (o_t)] olarak tanımlanabilir. Dolasıyla bu çalışmanın kurgusu göz önüne alındığında, DÖ ve MÖ modelleri arasında kullanabilecek en uygun model olduğu düşünülmektedir ve Şekil 2'de de görüleceği üzere, x_t sınıflandırılmak istenen değişkenleri temsil etmekle beraber, unutma kapısı (f_t) gelen bilginin bir sonraki adıma yani giriş kapısına (i_t) ne kadar aktartılacağını belirler, daha sonraki adım olan çıkış kapısında (o_t) ise sınıflama sınıflandırma tahmini yapılır (Şekil 2).



ŞEKİL 2: Uzun kısa dönem hafıza algoritması.

W∈ $\mathbb{R}^{hxd}$ ve $x_t \in \mathbb{R}^d$ olmak üzere

$$o_t = \sigma(W_o * [h_{t-1}, x_t]) \quad (7)$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t]) \quad (8)$$

$$h_t = tanh(W * [i_t * h_{t-1}, x_t]) \quad (9)$$

$$h_t = (1 - o_t) * h_{t-1} + o_t * h_t \quad (10)$$

Formülleri ile hesaplanır.[14]

Derin Sinir Ağları

Derin sinir ağları [deep neural networks (DNN)], temel olarak çok katmanlı algılayıcı [multi layer perceptron (MLP)] yapısına benzer ancak birden çok katman içeren bir yöntemdir. MLP yapısı gereği üç katmandan oluşur; giriş katmanı (input layer), gizli katman (hidden layer) ve çıkış katmanıdır (output layer). Sınıflama başarısını arttırmak adına, DNN mimarisinde gizli katman sayısı değişebilir, böylece MLP'ye göre daha kompleks bir yapı hâlini alarak daha yüksek sınıflama başarısı elde edilebilir.[15,16]

Model Değerlendirme Kriterleri

MÖ ve derin öğrenim algoritmalarının başarı performansları değerlendirilirken genel olarak, sensitivite, spesifite gibi birçok metrik kullanılmaktadır. Bu metriklerden genellikle doğru sınıflama oranı olan doğruluk (accuracy) değerleri ölçüt alınmaktadır.[17] Ancak bu çalışmada, doğru sınıflama oranın yanı sıra spesifite oranının kullanılması çalışmanın yapısı gereği daha uygun olacaktır. Spesifite, genel anlamda bir tanı testinin hasta-sağlıklı ayırımında gerçekten sağlıklı bireyleri yakalama oranı olarak tanımlanır. Bu çalışma, alternatif uç birleştirme var ve yok olarak ikili yapıda bir sınıflandırma mantığı ile kurgulanmıştır. Bu durumda, hastalık durumu olmadığı için alternatif uç bölgelerinin yakalanması önem taşımaktadır ve spesifite oranları sensitivite oranlarına göre daha fazla anlam arz etmektedir.

TABLO 2: Makine öğrenimi ve derin öğrenme modelleri validasyon performans sonuçları.

Model	Optimizasyon parametreleri
Rastgele orman	n_estimators=100, max_depth=10, random_state=42
Destek vektör makineleri	C=1.0, kernel='rbf', gamma='scale', random_state=42
Derin sinir ağları	Hidden layers: [128 (relu), 64 (relu), 1 (sigmoid)], epochs=10, batch_size=32
Uzun kısa dönem hafıza algoritması	units=64, activation='relu', epochs=10, batch_size=32

Tüm modeller için kullanılan en uygun optimizasyon parametreleri Tablo 2 ile verilmiştir.

BULGULAR

Örneklem genişliği bu çalışma için üretilen RNA transkripsiyon sayısı olarak ele alındığı göz önünde bulundurulmalıdır. Örneğin 200 örnek genişliği 100 hasta 100 kontrol transkripsiyonu olacak şekilde tasarlanmıştır ve rastgele 1.000 tane gen ve 300 sekanslama uzunluklarında olacak şekilde veriler üretilmiştir.

TABLO 3: Makine öğrenimi ve derin öğrenme modelleri validasyon performans sonuçları.

Örnek Genişliği	Model	Validasyon Doğruluk	Validasyon F1-Score	Validasyon Spesifite	Validasyon Sensitivite
25,00	DNN	0,67±0,14	0,70±0,14	0,74±0,13	0,73±0,14
25,00	LSTM	0,66±0,13	0,71±0,12	0,86±0,14	0,86±0,16
25,00	Rastgele orman	0,68±0,14	0,69±0,13	0,73±0,13	0,72±0,13
25,00	SVC	0,67±0,14	0,71±0,13	0,79±0,14	0,80±0,15
50,00	DNN	0,67±0,13	0,71±0,14	0,74±0,13	0,73±0,14
50,00	LSTM	0,67±0,13	0,72±0,12	0,85±0,14	0,86±0,16
50,00	Rastgele orman	0,68±0,14	0,70±0,14	0,74±0,13	0,73±0,13
50,00	SVC	0,67±0,13	0,72±0,13	0,82±0,14	0,82±0,15
100,00	DNN	0,70±0,13	0,73±0,15	0,77±0,13	0,76±0,16
100,00	LSTM	0,70±0,12	0,72±0,13	0,84±0,14	0,87±0,16
100,00	Rastgele orman	0,67±0,14	0,71±0,13	0,75±0,14	0,74±0,14
100,00	SVC	0,67±0,14	0,72±0,13	0,83±0,15	0,84±0,15
150,00	DNN	0,72±0,13	0,77±0,16	0,77±0,13	0,80±0,18
150,00	LSTM	0,71±0,13	0,74±0,14	0,82±0,12	0,88±0,16
150,00	Rastgele orman	0,69±0,13	0,72±0,15	0,78±0,14	0,77±0,16
150,00	SVC	0,70±0,12	0,73±0,12	0,84±0,14	0,86±0,16
200,00	DNN	0,75±0,13	0,76±0,17	0,78±0,12	0,84±0,19
200,00	LSTM	0,75±0,14	0,75±0,17	0,79±0,12	0,89±0,15
200,00	Rastgele orman	0,71±0,13	0,75±0,15	0,79±0,13	0,82±0,18
200,00	SVC	0,72±0,13	0,75±0,14	0,80±0,13	0,85±0,17

DNN: Derin sinir ağları; LSTM: Uzun kısa dönem hafıza algoritması; SVC: Destek vektör makineleri.

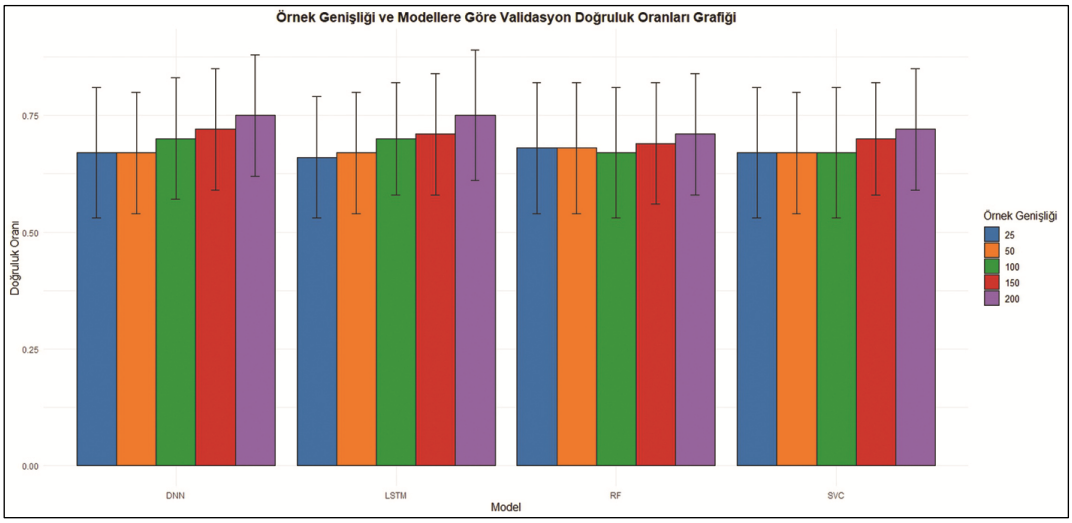
DNN modeli de RF modeline benzer doğru sınıflama performansı göstererek, 25 ve 50 örnek genişliklerinde hemen hemen aynı (0,67±0,14) sınıflama başarı oranlarını elde ederken, 200 örnek genişliğinde (0,75±0,13) ile LSTM modeliyle birlikte en yüksek başarı oranına sahiptir (Tablo 3).

Destek vektör makineleri [support vector machines (SVC)] için doğru sınıflama oranları göz önünde bulundurulursa, RF algoritmasına oldukça benzer sonuçlar elde ettiği görülmektir. Örneğin düşük örnek genişliklerinde RF ile hemen hemen benzer doğru sınıflama başarısı elde etmiştir (n=25, 0,67±0,14) (Tablo 3).

Tüm modellerin doğru sınıflama oranı performans kriteri bakımından hemen hemen benzer sonuçlar elde ettikleri gözlemlenmiştir. Örnek genişliğindeki değişim tüm modellerde doğru sınıflama başarısını artırmıştır (Tablo 3).

LSTM modeli, genellikle diğer modellere göre daha yüksek F1-Score, spesifite ve sensitivite değerlerine sahiptir. LSTM modelinin dengeli bir performans gösterdiği söylenebilir. RF modeli, genellikle yüksek örneklem büyüklüklerinde etkili bir şekilde çalışmaktadır. LSTM modeli için doğru sınıflama oranı en düşük 25 örnek genişliğinde iken, en yüksek 200 örnek genişliğindedir (0,66±0,13) (Tablo 3).

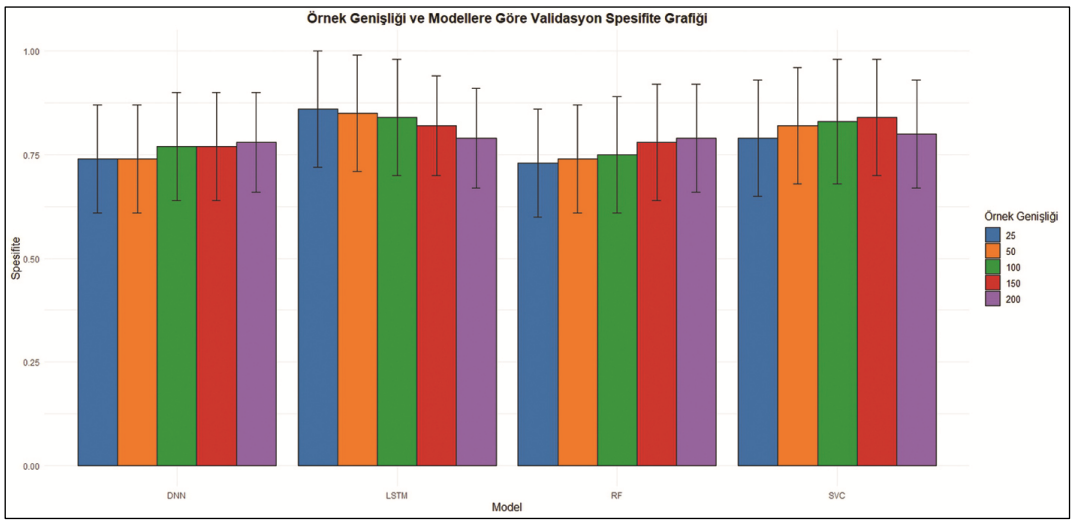
Sensitivite oranları incelenecek olursa, en yüksek sensitivite oranına 200 örnek genişliğinde LSTM modeli (0,89±0,15) elde ederken, en düşük sensitivite oranına RF modeli 25 örnek genişliğinde (0,72±0,13) elde etmiştir (Tablo 3).



DNN: Derin sinir ağları; LSTM: Uzun kısa dönem hafıza algoritması; RF: Rastgele orman, SVC: Destek vektör makineleri.

ŞEKİL 3: Makine öğrenimi ve derin öğrenme modelleri validasyon doğruluk performans sonuçları.

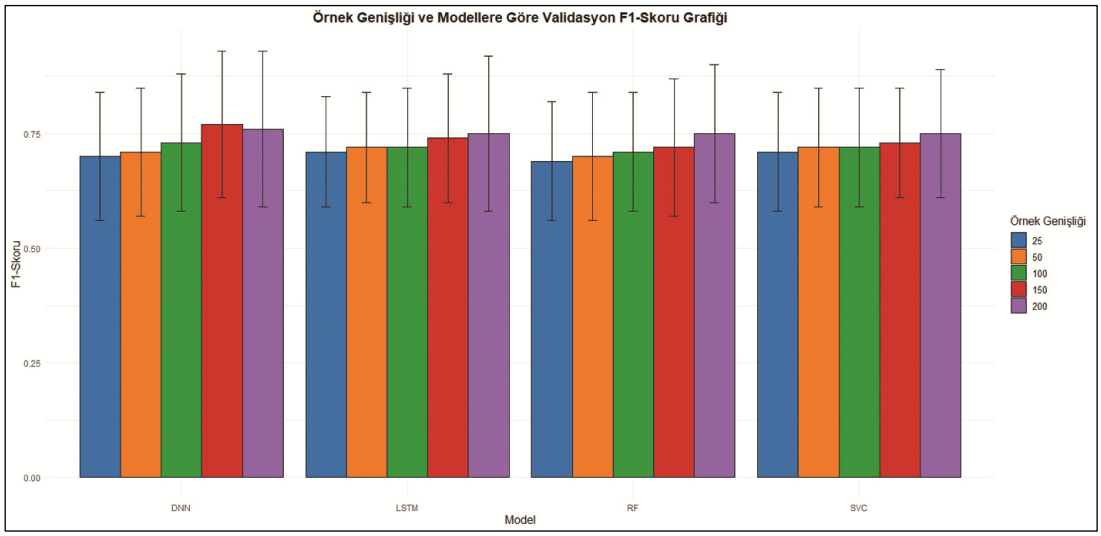
Örneklem büyüklüğü arttıkça modellerin performansının arttığı gözlemlenmiştir. Özellikle DNN ve LSTM modellerinin performansı, örneklem büyüklüğünün artmasıyla birlikte istikrarlı bir şekilde artmaktadır. RF modeli kendi içinde değerlendirildiğinde, 150 ve 200 örneklem büyüklüklerinde daha iyi performans gösterdiği gözlemlenmektedir ancak örnek genişliği 25 ve 50 olduğu zaman doğru sınıflama başarısı bakımından RF algoritması için bir fark görülmemektedir (Şekil 3).



DNN: Derin sinir ağları; LSTM: Uzun kısa dönem hafıza algoritması; RF: Rastgele orman; SVC: Destek vektör makineleri.

ŞEKİL 4: Makine öğrenimi ve derin öğrenme modelleri validasyon spesifite performans sonuçları.

Modellerin spesifite oranları incelenecek olursa, SVC modelinin 200 örnek genişliğinde diğer tüm modellere göre daha iyi sonuç elde ettiği görülmektedir. LSTM modelinin diğer tüm modellerden ayrıldığı kritik nokta ise düşük örnek genişliğinde dahi yüksek spesifite oranına sahip olmasıdır. RF algoritması için spesifite değerlerinin örnek genişliğinin artmasıyla birlikte istikrarlı bir şekilde arttığı görülmüştür (Şekil 4).



DNN: Derin sinir ağları; LSTM: Uzun kısa dönem hafıza algoritması; RF: Rastgele orman; SVC: Destek vektör makineleri.

ŞEKİL 5: Makine öğrenimi ve derin öğrenme modelleri validasyon F1 skoru performans sonuçları.

Sensitivite ve spesifiteyi bir arada kullanan F1 skoru bakımından tüm örnek genişliklerine göre modeller incelenecek olursa, DNN modelinin diğer tüm modellere göre en yüksek başarıyı 150 örnek genişliğinde elde ettiği görülmektedir (0,77±0,16) (Şekil 5).

TARTIŞMA

Elde edilen sonuçlara göre MÖ modellerini yalnızca doğru sınıflama performansı kriteri üzerinden değerlendirmek her zaman olumlu sonuçlar göstermemektedir. Tüm modellerin aşağı yukarı aynı doğru sınıflama oranı sonuçlar verdikleri gözlemlenmekle beraber, bu çalışmanın amacı doğrultusunda düşünülecek olursa, spesifite performans kriterinin dikkate alınması önem arz etmektir. Dolayısıyla LSTM ve SVC modellerinin yüksel doğruluk oranları yanı sıra yüksek F1 skor ve spesifite oranları da dikkat çekmektedir. Ayrıca SVC modeli için spesifite ve sensitivite değerlerinde örnek genişliğinin artmasıyla birlikte belirgin bir iyileşme gözlemlenmiştir. SVC modeli için benzeri bir durum FR modeli için de gözlemlenmektedir. MÖ algoritmaları, bu tür verileri hızla analiz edebilir ve alternatif uç birleştirmeleri tanımlayabilir.

Literatürdeki benzer çalışmalarda bu çalışmadaki bulgularla benzerlik göstermektedir. DNN kullanılarak yapılan bir çalışmada, 13240 gen ve 301835 alternatif uç bölgesi kullanılarak 0,86 ve üzeri doğru sınıflama başarısı elde edilmiştir.[18] LSTM algoritması kullanılarak, 13666 RNA sekanslama verisi kullanılarak, 0,92 F1 skoru elde edilmiştir.[19] Diğer bir çalışma da ise SVC modelinin alternatif uç bölgelerini yakalama konusunda oldukça yüksek sınıflama başarısı elde ettiği görülmüştür.[20]

Yüksek hesaplama gücü ve büyük boyutta depolama alanı gereksinimi nedeniyle alternatif uç birleştirme analizleri gibi yöntemlerin kişisel bilgisayarla uygulanması oldukça güçtür. Dolasıyla benzer yapıdaki simülasyon çalışmalarının kurgulanması oldukça güçtür. RNA sekanslama, tüm genom sekanslama, DNA metilasyon verileri gibi verilere ücretsiz olarak açık erişimli kaynakların çok az olması nedeniyle elde edilen simülasyon sonuçlarının birçok farklı hastalık için karşılaştırılması maliyet, zaman ve hesaplama yükü açısından oldukça güçtür.

Bu çalışmanın kısıtları olarak, alternatif uç birleştirme analizleri gibi yöntemler, yüksek hesaplama gücü ve büyük depolama alanı gerektirdiğinden, kişisel bilgisayarlarla uygulanması oldukça zor olduğu dile getirilebilir ve de benzer yapıdaki simülasyon çalışmaları kurgulamak da oldukça güçleşir. Diğer yandan, RNA sekanslama, tüm genom sekanslama ve DNA metilasyon verileri gibi verilere erişimin oldukça sınırlı olması,

elde edilen simülasyon sonuçlarının farklı hastalıklar için karşılaştırılmasını maliyet, zaman ve hesaplama yükü açısından zorlaştırır.

SONUÇ

Bu çalışma, alternatif uç birleştirmelerin belirlenmesi sürecinde, MÖ ve DÖ modellerinin etkili bir şekilde kullanılabileceğini ortaya koymaktadır. Bu çalışmanın literatürle de uyumlu olduğu görülmüştür. MÖ modelleri, özellikle düşük ve orta büyüklükteki veri setlerinde hızlı eğitim süreleri ve yeterli performans sunma avantajına sahiptir. Örneğin veri setinin genişliği arttıkça, SVC modelinin spesifite ve sensitivite değerlerinde istikrarlı bir iyileşme gözlemlenmiştir. Bu, MÖ yaklaşımının, özellikle daha az veri ile çalışılan durumlarda tercih edilebileceğini düşündürmektedir. Diğer yandan, DÖ modelleri, daha karmaşık örüntüleri öğrenme yetenekleri sayesinde büyük veri setlerinde daha yüksek doğruluk ve F1 skoru sağlayabilir. Özellikle, LSTM modelinin genişlik arttıkça yüksek performans sergilediği ve en yüksek F1 skorlarına ulaştığı gözlemlenmiştir. Bu, DÖ yaklaşımının, daha büyük ve karmaşık veri setleriyle çalışılan durumlarda daha uygun olabileceğini önermektedir. Bu bağlamda, veri setinin büyüklüğü, karmaşıklığı ve mevcut hesaplama kaynakları gibi faktörler dikkate alınarak tercih yapılmalıdır. Daha küçük ve basit veri setleriyle çalışılıyorsa, MÖ modelleri tercih edilebilirken, büyük ve karmaşık veri setleriyle çalışılıyorsa, DÖ modellerinin kullanılması daha uygun olabilir. Her iki yaklaşımın da avantajları ve sınırlılıkları göz önünde bulundurularak karar verilmelidir. Ayrıca hem MÖ hem de DÖ algoritmalarının performans kriterleri değerlendirilirken yalnızca doğru sınıflama başarısı oranın değil diğer metriklerin de çalışmanın hipotezi doğrultusunda dikkate alınması gerektiği önerilmektedir.

Finansal Kaynak

Bu çalışma sırasında, yapılan araştırma konusu ile ilgili doğrudan bağlantısı bulunan herhangi bir ilaç firmasından, tıbbi alet, gereç ve malzeme sağlayan ve/veya üreten bir firma veya herhangi bir ticari firmadan, çalışmanın değerlendirme sürecinde, çalışma ile ilgili verilecek kararı olumsuz etkileyebilecek maddi ve/veya manevi herhangi bir destek alınmamıştır.

Çıkar Çatışması

Bu çalışma ile ilgili olarak yazarların ve/veya aile bireylerinin çıkar çatışması potansiyeli olabilecek bilimsel ve tıbbi komite üyeliği veya üyeleri ile ilişkisi, danışmanlık, bilirkişilik, herhangi bir firmada çalışma durumu, hissedarlık ve benzer durumları yoktur.

Yazar Katkıları

Bu çalışma tamamen yazarın kendi eseri olup başka hiçbir yazar katkısı alınmamıştır.

KAYNAKLAR

1. Roca X, Sachidanandam R, Krainer AR. Determinants of the inherent strength of human 5' splice sites. RNA. 2005;11(5):683-98. [Crossref] [PubMed] [PMC]
2. Satam H, Joshi K, Mangrolia U, Waghoo S, Zaidi G, Rawool S, et al. Next-Generation Sequencing Technology: Current Trends and Advancements. Biology (Basel). 2023;12(7):997. [Crossref] [PubMed] [PMC]
3. Schmidt B, Hildebrandt A. Deep learning in next-generation sequencing. Drug Discov Today. 2021;26(1):173-80. [Crossref] [PubMed] [PMC]
4. Halperin RF, Hegde A, Lang JD, Raupach EA; C4RCD Research Group; Legendre C, Liang WS, LoRusso PM, Sekulic A, Sosman JA, Trent JM, et al. Improved methods for RNAseq-based alternative splicing analysis. Sci Rep. 2021;11(1):10740. [Crossref] [PubMed] [PMC]
5. Oubounyt M, Louadi Z, Tayara H, Chong KT. Deep learning models based on distributed feature representations for alternative splicing prediction. IEEE Access. 2018;6:58826-34. [Crossref]
6. Zhou K, Salamov A, Kuo A, Aerts AL, Kong X, Grigoriev IV. Alternative splicing acting as a bridge in evolution. Stem Cell Investig. 2015;2:19. [PubMed] [PMC]
7. Wickham H. Wiley interdisciplinary reviews: computational statistics. ggplot2. 2011;3(2):180-18. [Crossref]
8. Suthaharan S, Suthaharan S. Support vector machine. In: Suthaharan S, ed. Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning. 1st ed. New York: Springer; 2016. p.207-35. [Crossref]
9. Alpaydın E. Introduction to Machine Learning. 3rd ed. London: The MIT Press; 2004.

10. Gönen M, Alpaydın E. Multiple kernel learning algorithms. The Journal of Machine Learning Research. 2011;12:2211-68. [Link]
11. Strobl C, Zeileis A. Danger: High power!-exploring the statistical properties of a test for random forest variable importance. 2008. [Link]
12. Rigatti SJ. Random forest. J Insur Med. 2017;47(1):31-9. [Crossref] [PubMed]
13. Yu Y, Si X, Hu C, Zhang J. A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput. 2019;31(7):1235-70. [Crossref] [PubMed]
14. Duan J, Gong Y, Luo J, Zhao Z. Air-quality prediction based on the ARIMA-CNN-LSTM combination model optimized by dung beetle optimizer. Sci Rep. 2023;13(1):12127. [Crossref] [PubMed] [PMC]
15. Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, et al. Evolving deep neural networks. In: Kozma R, Alippi C, Morabito FC, eds. Artificial Intelligence in the Age of Neural Networks and Brain Computing. 1st ed. London, United Kingdom: Academic Press; 2019. p.293-312. [Crossref]
16. Ser G, Bati CT. Derin sinir ağları ile en iyi modelin belirlenmesi: mantar verileri üzerine keras uygulaması [Determining the best model with deep neural networks: Keras application on mushroom data]. Yuzuncu Yıl University Journal of Agricultural Sciences. 2019;29(3):406-17. [Crossref]
17. Orozco-Arias S, Piña JS, Tabares-Soto R, Castillo-Ossa LF, Guyot R, Isaza G. Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements. Processes. 2020;8(6):638. [Crossref]
18. Strauch Y, Lord J, Niranjan M, Baralle D. CI-SpliceAI-Improving machine learning predictions of disease causing splicing variants using curated alternative splice sites. PLoS One. 2022;17(6):e0269159. [Crossref] [PubMed] [PMC]
19. Regan K, Saghafi A, Li Z. Splice junction identification using long short-term memory neural networks. Curr Genomics. 2021;22(5):384-390. [Crossref] [PubMed] [PMC]
20. Baten A, Chang B, Halgamuge S, Li J. Splice site identification using probabilistic parameters and SVM classification. BMC Bioinformatics. 2006;7(Suppl 5):S15. [Crossref] [PubMed] [PMC]