PromoterME: یک روش شناسایی ترویج کننده ها با استفاده از شبکه عصبی ترکیبی و استخراج ویژگی از توالی DNA بر اساس شبکه عصبی پرسپترون چند لایه ای

نیلوفر آقایی ابیانه[1] و مهدی رحیمی[2]

[1] دپارتمان الگوریتم ها و محاسبات، دانشکده فنی، دانشگاه تهران niloofaraghaie@ut.ac.ir

[2] دپارتمان الگوریتم ها و محاسبات، دانشکده فنی، دانشگاه تهران Mehdi.rahimi@ut.ac.ir

چکیده

شناسایی ترویج کننده[1] ها برای تنظیمات بیان ژن[2] بسیار مهم می باشند. اگرچه الگوریتم هایی برای شناسایی ترویج کننده ها ایجاد شده است؛ اما همچنان توسعه یک الگوریتم با دقت بالا چالشی بزرگ محسوب می شود. در این مقاله، یک الگوریتم برای شناسایی ترویج کننده ها ارائه می شود که به آن PromoterME گفته می شود. در این الگوریتم تنها از ویژگی DNA عددی شده[3] برای شناسایی ترویج کننده ها استفاده می شود. و هر یک از چهار حرف A، C، Gو T به عنوان یک ویژگی در نظر گرفته می شود. که این موضوع سبب می شود که فضای ویژگی خیلی بزرگ شود، لذا برای کاهش فضای ویژگی از شبکه عصبی پرسپترون چند لایه ای استفاده می شود. در آخر با استفاده از روال یادگیری *Adaboost* عمل شناسایی ترویج کننده ها انجام می شود.

برای پیاده سازی این الگوریتم از پایگاه داده های EPD[4] استفاده شده است.

۱. مقدمه

ترویج کننده ها قسمتی از توالی DNA می باشند که منجر به انجام عمل رونویسی[5] از ژن می شود. هر ترویج کننده دارای مجموعه های مشخصه از توالی های کوچک حفاظت شده می باشند؛ این توالی ها همان ویژگی های ساختاری ترویج کننده ها می باشند. با این وجود، این ویژگی ها در همه ی ترویج کننده ها موجود نیست. از ویژگی های ساختاری ترویج کننده ها می توان به TATA-box که در نواحی غنی شده از AT قرار دارند، نواحی غنی شده از CpG و CAAT-box اشاره

[1] promoter
[2] Gene regulation
[3] Digitized DNA
[4] Eukaryotic Promoter Database
[5] transcription

کرد. اگرچه بعضی از ترویج کننده ها فاقد TATA-box می باشند که به آنها ترویج کننده های فاقد TATA-box[1] گفته می شود. از این ویژگی ها می توان برای شناسایی نواحی ترویج کننده ها و غیر ترویج کننده ها استفاده کرد[۱][۲][۳] [۴].

الگوریتم های متعددی برای شناسایی نواحی ترویج کننده ها ارائه شده است مثلاً *Dragon Promoter Finder (DPF)*، *PromoterExplorer* ، *PromoterInspector*.

از یک دیدگاه می توان این الگوریتم ها را به دو دسته ی بر اساس ویژگی های مفهومی مانند الگوریتم PromoterInspector و بر اساس ویژگی های ساختار های مانند PromoterExplorer تقسیم بندی کرد[۵].

هر یک از این الگوریتم ها از شناسایی الگو های[2] مختلفی برای این مسئله طبقه بندی و همچنین از ویژگی های مختلفی استفاده می کنند، مثلاً می توان به شبکه عصبی، تحلیل های خطی و مربعی[3]، مدل مارکوف[4] و تحلیل مولفه مستقل[5] اشاره کرد. [۶] . درDPF هر توالی از پنج نوکلئوتید پشت سر هم به عنوان فضای ویژگی در نظر گرفته می شود[۷]. در *PromoterInspctor* دو مجموعه از الیگونوکلئوتیدها به ترتیب مرتبط با ترویج کننده ها و غیر ترویج کننده ها در نظر گرفته می شود؛ همچنین جایگزین های متعددی در مکان های چندگانه معرفی می شوند. تعداد جایگزین ها و طول عناصر در الیگونوکلئوتیدها برای طبقه بندی بهینه می شوند و سپس برای تست کردن مورد استفاده قرار می گیرند[۸].

در *PromoterExplorer* توزیع محلی توالی های پنج تایی از نوکلئوتید ها، نواحی CpG و DNA عددی شده به عنوان فضای ویژگی در نظر گرفته می شوند. سپس با کمک الگوریتم طبقه بندی *AdaBoost* عمل طبقه بندی انجام می شود[۶][۹].

در اینجا کارایی الگوریتم ارائه شده در این مقاله روی پایگاه داده EPD[6] محاسبه می شود. در روش ارائه شده در این مقاله تنها از ویژگی DNA عددی شده استفاده می شود و هر یک از باز های A، C، Gو T و به عنوان یک ویژگی در نظر گرفته می شوند؛ که این به نوبه ی خود منجر به یک فضای ویژگی خیلی بزرگ می شود. برای حل این مشکل، با کمک شبکه عصبی چند لایه ای ابتدا فضای ویژگی کاهش داده می شود؛ سپس با استفاده از مدل ترکیبی طبقه بندی این مسئله دو کلاسه انجام داده می شود. در این مقاله از مدل های ترکیبی

[1] TATA-box less promoter
[2] Pattern recognition
[3] Linear and quadratic discriminant analyses
[4] Markov model
[5] Independent component analyses
[6] Eukaryotic Promoter Database (EPD)

*AdaBoostM*۱، *LogitBoost* و *GentleBoost* استفاده شده است.

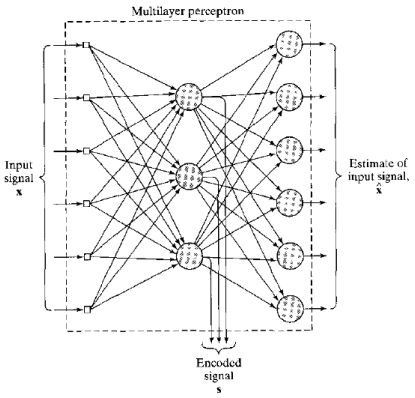
این مقاله به صورت زیر سازماندهی شده است.

۲. استخراج ویژگی از توالی DNA

همانطور که گفته شد در این مقاله از ویژگی DNA عددی شده استفاده شده است. در اینجا هر یک از باز های A، C، Gو T به ترتیب معادل با ۰۰، ۰۱، ۱۰ و ۱۱ قرار داده شده، و نهایتاً به عنوان یک ویژگی در نظر گرفته می شوند [۱۰]. بنابراین اگر l طول توالی DNA باشد، فضای ویژگی 2^l می شود. برای کاهش فضای ویژگی از شبکه عصبی پرسپترون سه لایه ای استفاده می شود. برای این کار در این شبکه عصبی تعداد نرون های لایه ورودی و خروجی برابر با 2^l قرار می گیرند و تعداد نرون های لایه میانی برابر با طول بردار ویژگی مورد انتظار در نظر گرفته می شوند. شکل ۱ این موضوع را نشان می دهد. به ازای هر نمونه آموزش بردار ورودی و خروجی یکسان قرار داده می شوند و وزن های این شبکه عصبی به صورت مد الگو[۱] طبق قاعده ی دلتا به روز می شوند.

$$w(n+1) = w(n) + \eta . \varepsilon . f' . X$$

که $w(n)$ وزن ها در لحظه ی n ام، η نرخ یادگیری، ε خطا، f' مشتق تابع f و X بردار ورودی می باشند، فرآیند به روز رسانی وزن ها تا زمانیکه خطای مجذور مربعات[۲] به مقداری مشخص برسد، ادامه می یابد. با این کار با داشتن وزن های لایه ورودی به میانی (با فرض اینکه شبکه عصبی ۳ لایه ای باشد) می توان برای هر نمونه با پیش پردازش روی نمونه، ابعاد آن را به مقدار مورد نظر کاهش داد[۱۱].



شکل ۱ – ساختار شبکه عصبی پرسپترون سه لایه برای استخراج ویژگی

در اینجا تعداد نرون های لایه میانی برابر با ۸ قرار میگیرد. در نتیجه پس از اینکه توالی DNA ورودی به صورت عددی شد فضای ویژگی برابر با 2^l می شود که با این عمل استخراج به ۸ کاهش می یابد.

[۱] Pattern mode

[۲] Minimum Square Error (MSE)

۳. آموزش طبقه بند ها برای عمل طبقه بندی

Boosting یک الگوریتم یادگیری با نظارت[۱] است که برای بهبود صحت تصمیم گیری هر الگوریتم یادگیری به کار می رود. در این الگوریتم، یک طبقه بند که دقتش بر اساس مجموعه آموزش است بزرگتر از کارایی در یک حالت متوسط ایجاد می شود، و سپس طبقه بند های جدید به منظور ایجاد یک شکل ترکیبی اضافه می شوند؛ به طوریکه حاصل از این طبقه بندها دارای دقت بالایی باشد [۱۲][۱۳]. در این حالت گفته می شود که کارایی طبقه بندی ارتقا[۲] یافته است. به طور کلی آموزش های الگوریتم های طبقه بند پشت سر هم به همراه زیر مجموعه ای از کل داده آموزش که از همه با اهمیت تر است به مجموعه ای از طبقه بند های فعلی داده می شود.

Adaboost یک الگوریتم boosting است که یک یادگیرنده ی ضعیف[۳] را روی داده آموزش که هر بارکمی تغییر کرده است چندین بار اجرا می کند، و سپس فرضیات را به منظور به دست آوردن دقت بالاتری از فرضیاتی که یادگیرنده های ضعیف دارند با هم ترکیب می کند. ایده اصلی *Adaboost* به این صورت است که هر نمونه ای از مجموعه آموزش برای تمایز در گام های آموزش متفاوت نقش متفاوتی را ایفا کنند. این نمونه ها که به آسانی شناسایی می شوند باید در آموزش به صورت زیر کمتر در نظر گرفته شوند؛ تا زمانیکه نمونه ای اشتباه طبقه بندی می شود باید در مراحل بعدی به آن بیشتر توجه شود. برای این منظور یادگیرنده ضعیف باید روی نمونه های با اهمیت تر یا "سختر" تمرکز کند. اهمیت نمونه ها با وزن نشان داده می شود. در ابتدا همه وزن ها برابرند. سپس وزن ها هر بار باید بر اساس نتایج طبقه بندی تغییر کنند. تصمیم گیری نهایی ترکیبی از تصمیمات در هر مرحله است[۱۲] [۱۳].

اگر الگوریتم *AdaBoost* را به صورت یک مدل جمعی عام[۴] در نظر گرفته شود و سپس تابع هزینه ی[۵] رگرسیون لگاریتمی[۶] بر آن اعمال شود، الگوریتم *LogitBoost* حاصل می شود. به عبارت دیگر *LogitBoost* همان الگوریتم *AdaBoost* که تابع هزینه ی آن رگرسیون لگاریتمی باشد.

GentleBoost نیز نسخه ای تغییر یافته از *AdaBoost* است [۱۳].

---

۱ Supervised Learning Algorithm
۲ boosted
۳ Weak learner
۴ generalized additive model
۵ cost function
۶ logistic reqression

۴. نتایج آزمایشگاهی

پس از فاز استخراج ویژگی یک مجموعه داده۱۰بعدی ایجاد می شود. در ادامه این مجموعه داده۱۰بعدی برای طبقه بند ترکیبی به کار می رود.

در این آزمایش ها به تعداد ۱۰۰۰ نمونه از توالی ترویج کننده ها از پایگاه داده EPD و ۱۰۰۰ نمونه از توالی غیر ترویج کننده ها انتخاب شده است، که ابتدا این ۲۰۰۰ نمونه در فاز استخراج ویژگی به یک مجموعه داده ای ۲۰۰۰ نمونه ای با ابعاد ۸ تبدیل می شود. پس از این فاز ویژگی ها به صورت عددی می باشند. از طرفی در هر آزمایش به میزان ۰/۸ از کل مجموعه داده ای برای آموزش و ۰/۲ با قیمانده برای تست و ارزیابی کارایی برنامه به کار می رود. اگرچه در هر آزمایش میزان داده های آموزش و تست ثابت در نظر گرفته می شود، اما این نمونه ها به صورت تصادفی انتخاب می شوند که بنابراین منجر به نتایج مختلف می شود. در این آزمایش ها *learners* نوع *Tree* و سه نوع *method* (*AdaBoostM*۱، LogitBoost و GentleBoost) مورد آزمایش قرار گرفته اند. همچنین هر آزمایش برای ۳ مقدار نرخ یادگیری ۰,۱، ۰,۵ و ۰,۹ روی داده های آموزش و تست ثابت بررسی می شود.

در هر آزمایش خطای جایگذاری[۱] بررسی می شود. تخمین این خطا براساس تفاوت بین مقادیر پیش بینی شده از یک مدل آزمایش و مقادیر مشاهده شده در آزمایش ها اندازه گیری می شود. همچنین معیار های حساسیت[۲] (S_n) و خاصیت (S_p) محاسبه می شوند. این معیار ها برابرند با:

$$S_n = \frac{TP}{TP + FN}$$

$$S_p = \frac{TP}{TP + Fp}$$

که *TP*، شناسایی نواحی به عنوان ترویج کننده هایی که واقعاً ترویج کننده میباشند؛ *FN*، شناسایی نواحی به عنوان غیر ترویج کننده هایی که واقعاً ترویج کننده میباشند؛ *FP*، شناسایی نواحی به عنوان ترویج کننده هایی که واقعاً ترویج کننده نمی باشند.

در این آزمایش ها *learners* نوع *Tree* و سه نوع *method* (*AdaBoostM*۱، LogitBoost و GentleBoost) مورد آزمایش قرار گرفته اند.

برای شبیه سازی این طبقه بندی ترکیبی از تابع *fitensemble* در محیط متلب استفاده شده است. این تابع یک مدل ترکیبی را برای پیش بینی پاسخ به داده را می سازد. ساختار نحوی این تابع در محیط متلب به صورت زیر است.

Ens = fitensemble (X,Y,method,nlearn,learners)

---

[۱] Resubstitution Error

[۲] sensivity

و پارامترهای تابع *fitensemble* به صورت زیر است. این ماشین بر اساس مدل های لیست شده در پارامتر *learners* ایجاد می شود.

توصیف پارامترهای اصلی

X: ماتریسی است که هر ستون آن مشخص کننده ی یک متغیر (ویژگی) است و هر سطر آن یک نمونه از مجموعه داده ای را نشان می دهد.

Y: در مسائل طبقه بندی مشخص کننده ی یک بردار ستونی است که یک کلاس را به ازای هر نمونه از مجموعه داده ای مشخص می کند.

method: یک رشته ی حساس به کوچک بزرگی حروف است که یکی از مقادیر زیر را بسته به تعداد کلاس ها می گیرد:

برای طبقه بندی دو کلاسه

- *AdaBoostM۱*
- *LogitBoost*
- *GentleBoost*
- *RobustBoost*
- *Bag*
- *Subspace*

برای طبقه بندی سه کلاسه و بیشتر

- *AdaBoostM۲*
- *Bag*
- *Subsapce*

nlearn: یک عدد صحیح مثبت است که تعداد چرخه های یادگیری[۱] ترکیبی را نشان می دهد. در هر چرخه ی یادگیری تابع *fitensemble* بر روی همه ی الگوهای یادگیری لیست شده در پارامتر عبور می کند و یک یادگیر ضعیف را برای هر الگو آموزش می دهد. تعداد کل یادگیر های آموزش دیده برابر حاصل ضرب *nlearn* در تعداد *learners* است.

s : مشخص کننده ی نوع یادگیرهای ضعیف است. سه نوع از این یادگیرها که ما نیز در آزمایش ها از آن استفاده می شود. عبارت اند از

- *Discriminant*
- *KNN*
- *Tree*

در عین حال تعدادی پارامتر اختیاری ممکن برای همه و در برخی موارد برای انواع خاصی از *method* ها وجود دارد. یدر اینجا یکی از پارامترهایی که مورد آزمایش قرار گرفته *learning rate* است. در محیط متلب به صورت پیش فرض مقدار آن ۱ است[۱۵] [۱۴] .

در این مقاله آزمایش های انجام شده به صورت زیر است.

[۱] Learning cycle

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۵۶/۶۶۶۷	۴۶/۶۶۶۷	۳۶/۶۶۶۷
LogitBoost	۴۶/۶۶۶۷	۴۳/۳۳۳۳	۴۳/۳۳۳۳
GentleBoost	۴۰	۴۰	۴۰

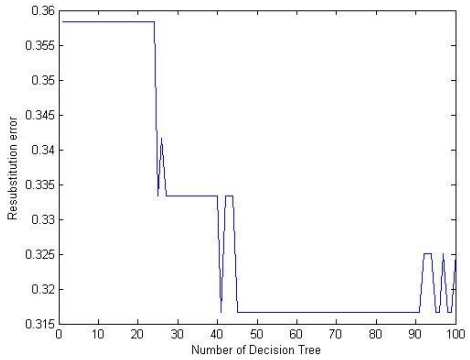
جدول ۱ – مقادیر کارایی در آزمایش اول به ازای LR های ۰/۱، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۷۰۸۳	۰/۷۰۰۰	۰/۶۴۷۱
LogitBoost	۰/۷۰۰۰	۰/۶۸۴۲	۰/۶۸۴۲
GentleBoost	۰/۵۲۱۷	۰/۵۲۱۷	۰/۶۳۱۶

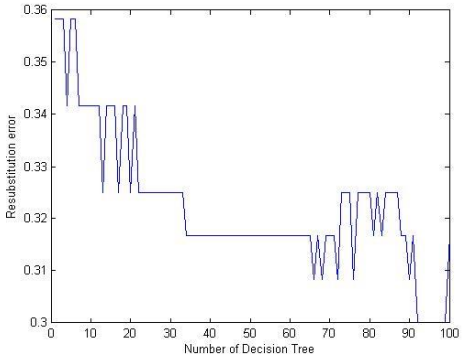
جدول ۲ – مقادیر S_n در آزمایش اول به ازای LR های ۰/۱، ۰/۵ و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۷۳۹۱	۰/۵۸۳۳	۰/۴۵۸۳
LogitBoost	۰/۵۸۳۳	۰/۵۴۱۷	۰/۵۴۱۷
GentleBoost	۰/۶۳۱۶	۰/۶۳۱۶	۰/۵۲۱۷

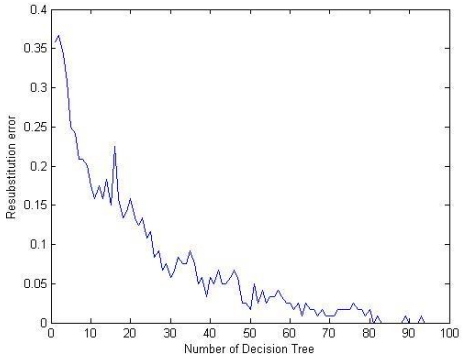
جدول ۳ – مقادیر S_p در آزمایش اول به ازای LR های ۰/۱، ۰/۵ و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost



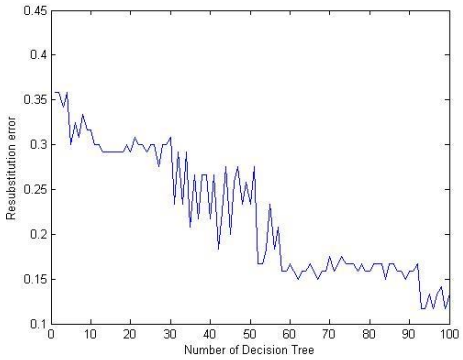
شکل ۲- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۱ و روش AdaBoostM۱



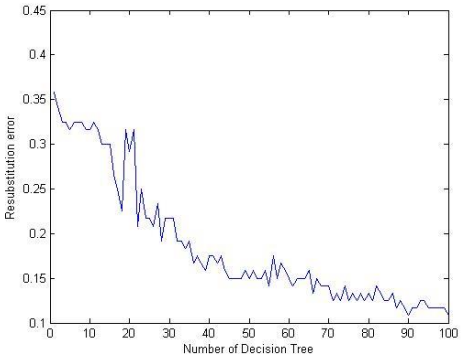
شکل ۳- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۱ و روش LogitBoost۱



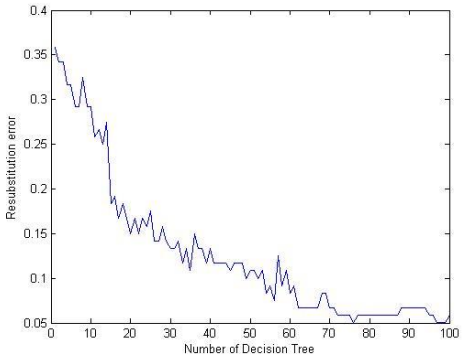
شکل ۴- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۱ و روش GentleBoost



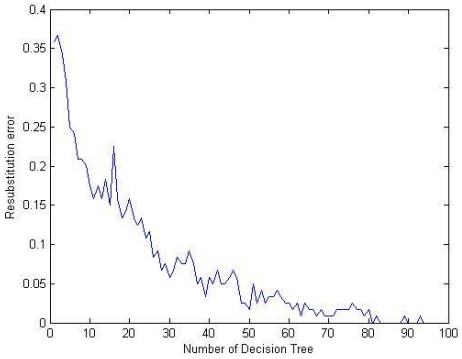
شکل ۵- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۵ و روش AdaBoostM۱



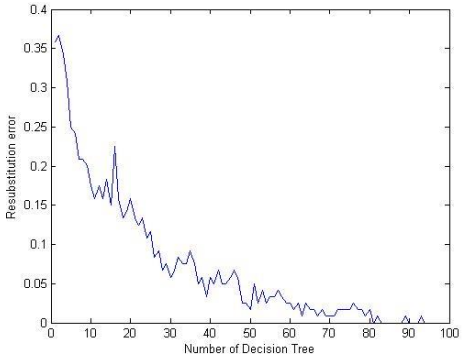
شکل ۶- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۵ و روش LogitBoost۱



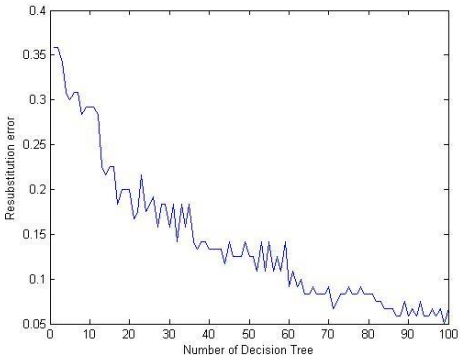
شکل ۹- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۹ و روش LogitBoost۱



شکل۷- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۵ و روش GentleBoost



شکل ۱۰ - نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۹ و روش GentleBoost



شکل ۸- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش اول با LR = ۰/۹ و روش AdaBoostM۱

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۵۰	۳۶/۶۶۶۷	۳۳/۳۳۳۳
LogitBoost	۵۰	۴۰	۳۳/۳۳۳۳
GentleBoost	۳۰	۳۰	۳۰

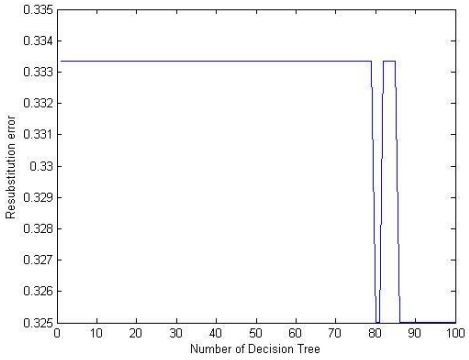
جدول ۴ – مقادیر کارایی در آزمایش دوم به ازای LR های ۰/۱، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۵۷۶۹	۰/۵۰۰۰	۰/۵۲۶۳
LogitBoost	۰/۵۷۶۹	۰/۵۴۵۵	۰/۵۰۰۰
GentleBoost	۰/۴۷۳۷	۰/۴۷۳۷	۰/۴۷۳۷

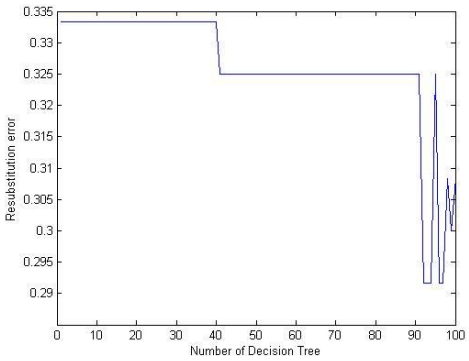
جدول ۵ – مقادیر S_n در آزمایش دوم به ازای LR های ۰/۱، ۰/۵ و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۷۳۹۱	۰/۵۸۳۳	۰/۴۵۸۳
LogitBoost	۰/۵۸۳۳	۰/۵۴۱۷	۰/۵۴۱۷
GentleBoost	۰/۶۳۱۶	۰/۶۳۱۶	۰/۵۲۱۷

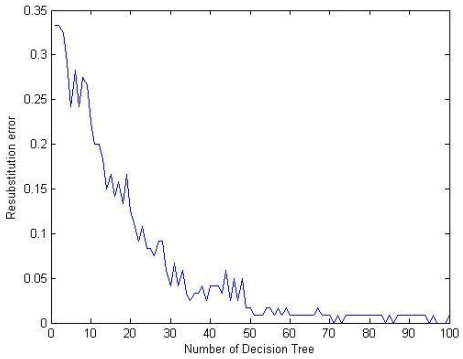
جدول ۶ – مقادیر S_p در آزمایش دوم به ازای LR های ۰/۱ ، ۰/۵ و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost



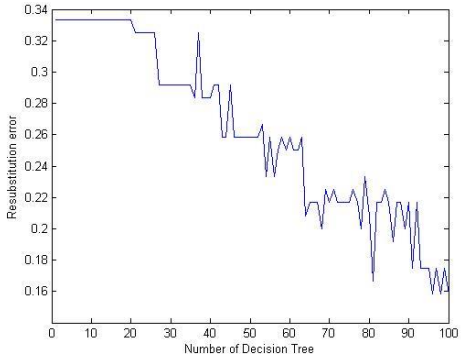
شکل ۱۱- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۱ و روش AdaBoostM۱



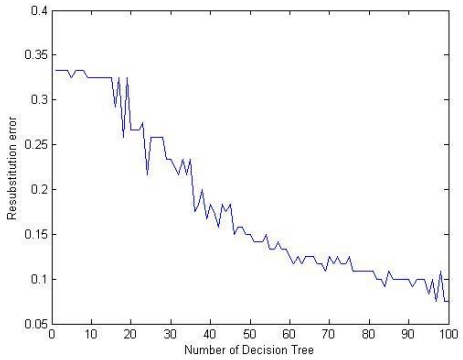
شکل ۱۲- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۱ و روش LogitBoost۱



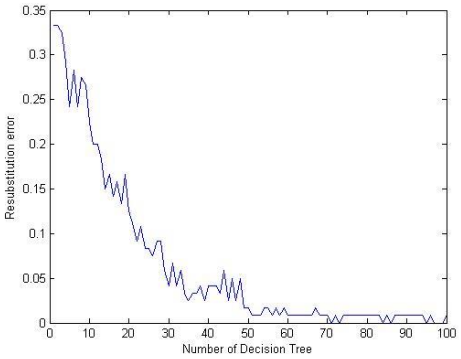
شکل۱۳- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۱ و روش GentleBoost



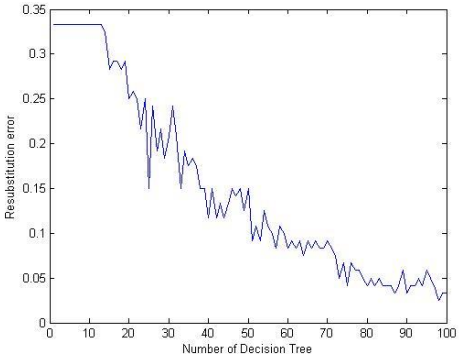
شکل ۱۴- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۵ و روش AdaBoostM۱



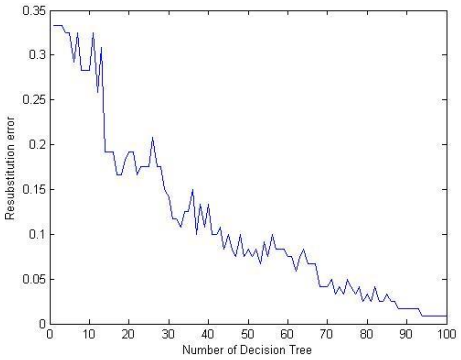
شکل ۱۵- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۵ و روش LogitBoost۱



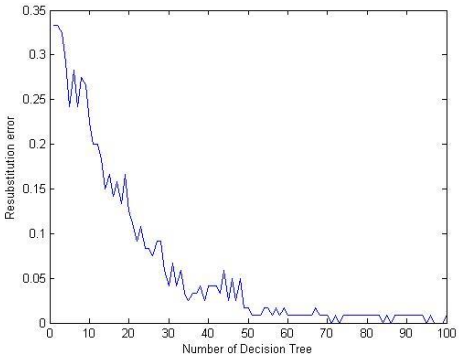
شکل۱۶- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۵ و روش GentleBoost



شکل ۱۷- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۹ و روش AdaBoostM۱



شکل ۱۸- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۹ و روش LogitBoost۱



شکل ۱۹ - نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش دوم با LR = ۰/۹ و روش GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۵۰	۴۶/۶۶۶۷	۵۳/۳۳۳۳
LogitBoost	۵۰	۴۶/۶۶۶۷	۴۳/۳۳۳۳
GentleBoost	۵۰	۵۰	۵۰

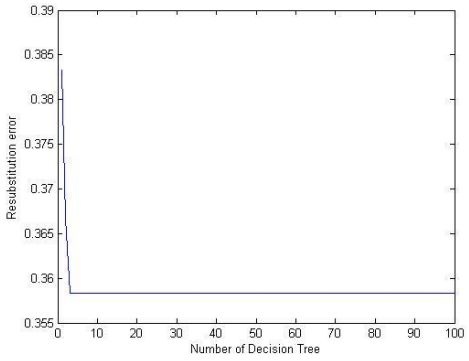
جدول ۷ – مقادیر کارایی در آزمایش سوم به ازای LR های ۰/۱، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۷۵۰۰	۰/۷۳۶۸	۰/۸۰۰۰
LogitBoost	۰/۷۵۰۰	۰/۷۷۷۸	۰/۷۶۴۷
GentleBoost	۰/۷۵۰۰	۰/۶۰۰۰	۰/۶۰۰۰

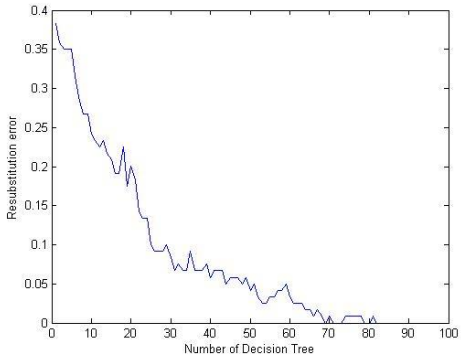
جدول ۸ – مقادیر S_n در آزمایش سوم به ازای LR های ۰/۱، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۶۰۰۰	۰/۵۶۰۰	۰/۶۱۵۴
LogitBoost	۰/۶۰۰۰	۰/۵۳۸۵	۰/۵۰۰۰
GentleBoost	۰/۶۰۰۰	۰/۷۵۰۰	۰/۷۵۰۰

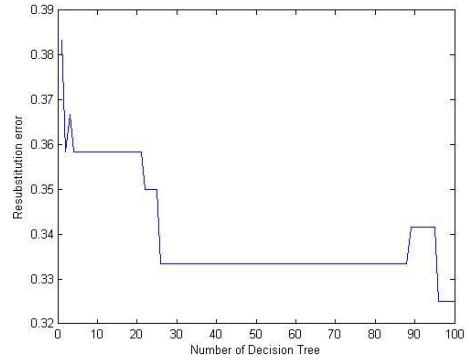
جدول ۹ – مقادیر S_p در آزمایش سوم به ازای LR های ۰/۱، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost



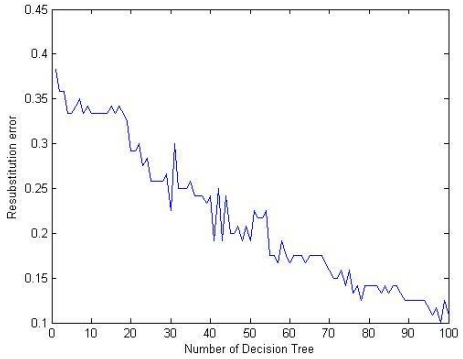
شکل ۲۰- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۱ و روش AdaBoostM۱



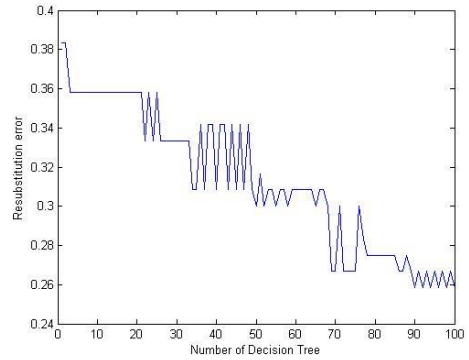
شکل ۲۳- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۵ و روش AdaBoostM۱



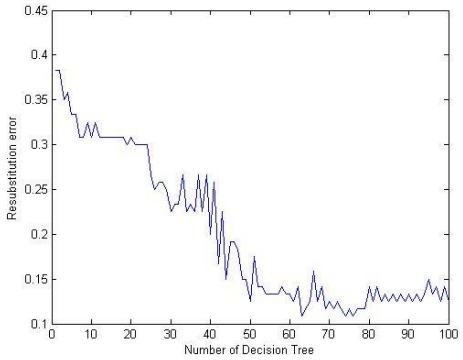
شکل ۲۱- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۱ و روش LogitBoost۱



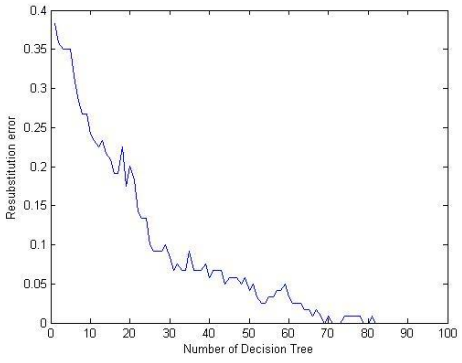
شکل ۲۴- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۵ و روش LogitBoost۱



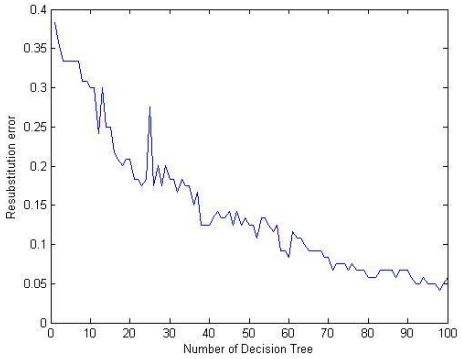
شکل ۲۲- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۱ و روش GentleBoost



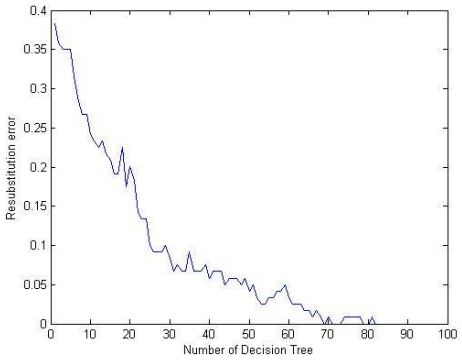
شکل ۲۵- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۵ و روش GentleBoost



شکل ۲۶- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۹ و روش AdaBoostM۱



شکل ۲۷- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۹ و روش LogitBoost۱



شکل ۲۸ - نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۹ و روش GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۷۳/۳۳۳۳	۶۳/۳۳۳۳	۵۶/۶۶۶۷
LogitBoost	۷۳/۳۳۳۳	۶۶/۶۶۶۷	۵۳/۳۳۳۳
GentleBoost	۴۶/۶۶۶۷	۴۶/۶۶۶۷	۴۶/۶۶۶۷

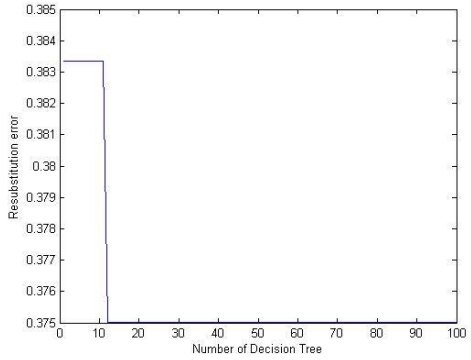
جدول ۱۰ – مقادیر کارایی در آزمایش چهارم به ازای LR های ۰/۱ ، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۰/۷۳۳۳	۰/۷۹۱۷	۰/۷۳۹۱
LogitBoost	۰/۷۸۵۷	۰/۸۳۳۳	۰/۷۲۷۳
GentleBoost	۰/۶۶۶۷	۰/۶۶۶۷	۰/۶۶۶۷

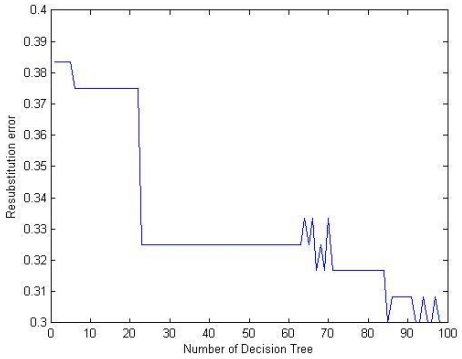
جدول ۱۱ – مقادیر S_n در آزمایش چهارم به ازای LR های ۰/۱ ، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost

	LR = ./۱	LR = ۰/۵	LR = ۰/۹
AdaBoostM۱	۱	۰/۷۶۰۰	۰/۷۰۸۳
LogitBoost	۰/۹۱۶۷	۰/۷۶۹۲	۰/۶۶۶۷
GentleBoost	۰/۶۰۸۷	۰/۶۰۸۷	۰/۶۰۸۷

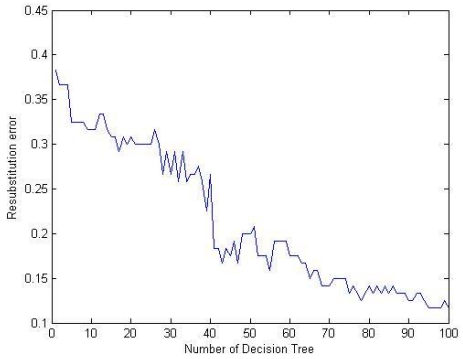
جدول ۱۲ – مقادیر S_p در آزمایش چهارم به ازای LR های ۰/۱ ، ۰/۵و ۰/۹ با روش های AdaBoostM۱، LogitBoost، GentleBoost



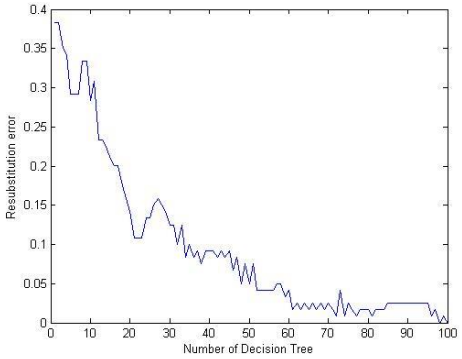
شکل ۲۹- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش سوم با LR = ۰/۱ و روش AdaBoostM۱



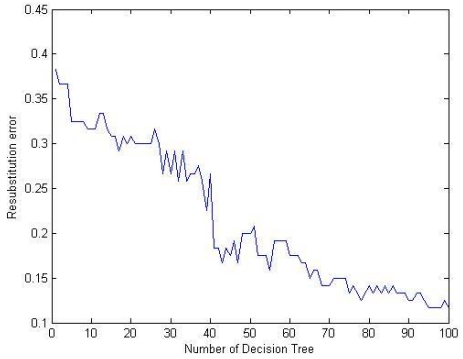
شکل ۳۰- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۱ و روش LogitBoost۱



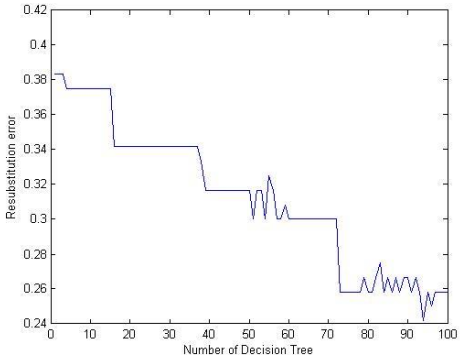
شکل ۳۳- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۵ و روش LogitBoost۱



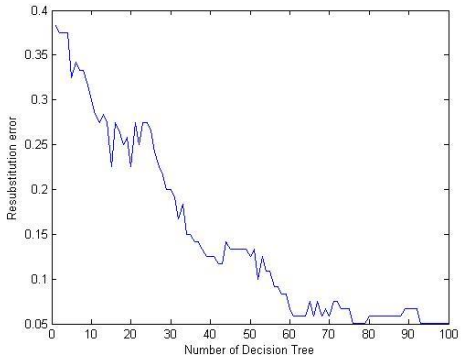
شکل ۳۱- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۱ و روش GentleBoost



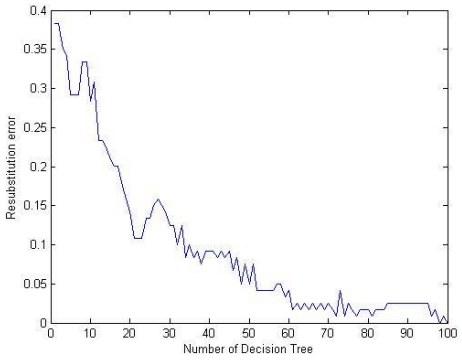
شکل ۳۴- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۵ و روش GentleBoost



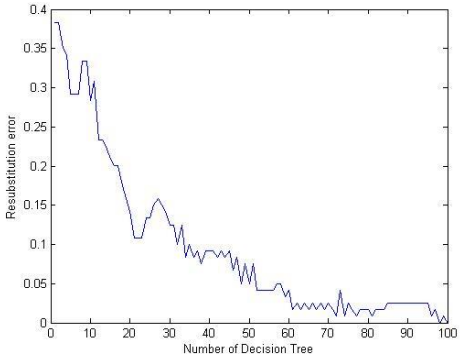
شکل ۳۲- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۵ و روش AdaBoostM۱



شکل ۳۵- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۹ و روش AdaBoostM۱



شکل ۳۶- نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۹ و روش LogitBoost۱



شکل ۳۷ - نمودار خطای جایگزینی به ازای تعداد درخت ها در آزمایش چهارم با LR = ۰/۹ و روش GentleBoost

۵. نتیجه گیریی

با توجه به نتایج حاصل از چهار بار آزمایش که در جداول ۱ تا ۱۲ و شکل های ۲ تا ۳۷ آمده است .مشخص می شود که الگوریتم PromoterME نسبت به الگوریتم های Dragon Promoter Finder، PromoterInspector، و PromoterExplorer از کارایی پایین خوردار است. که البته این نسخه نهایی آن نمی باشد و نیاز به بررسی و انجام آزمایش های بیشتر بخصوص در قسمت استخراج ویژگی است. در واقع این مقاله تنها عرفی از الگوریتمی که به زودی نسخه ی نهایی آن منتشر خواهد شد می باشد. این الگوریتم نیاز به بهینه سازی در قسمت های مختلف از جمله در استخراج اطلاعات، تعیین پارامتر های بهینه برای شبکه های عصبی و... است. که به زودی انجام خواهد شد.

۶. منابع

[۱]Benjamin Lewin(۲۰۰۷),Genes,edition ۹, Jones and Bartlett Publishers.

[۲] James W. Fickett and Artemis G. Hatzigeorgiou(۱۹۹۷) Eukaryotic Promoter Recognition, Genome Res ۷: ۸۶۱-۸۷۸.

[۳] Anders Gorm Pedersen, Pierre Baldi, Yves Chauvin, and Søren Brunak(۱۹۹۹),The Biology of Eukaryotic Promoter Prediction—a Review.

[۴] Aditi Kanhere and Manju Bansal(۲۰۰۵), A novel method for prokaryotic promoter prediction based on DNA stability, BMC Bioinformatics ۲۰۰۵, ۶:۱ doi:۱۰,۱۱۸۶/۱۴۷۱-۲۱۰۵-۶-۱

[۵] Jia Zeng, Shanfeng Zhu and Hong Yan(۲۰۰۹), Towards accurate human promoter recognition: a review of currently used sequence features and classification methods, BRIEFINGS IN BIOINFORMATICS. VOL ۱۰. NO ۵. ۴۹۸ -۵۰۸

[۶] Xudong Xie,Shuanhu Wu,Kin-Man Lam and Hong Yan(۲۰۰۶), PromoterExplorer: an effective promoter identification method based on the AdaBoost algorithm.bioinformatic ,Vol. ۲۲ no. ۲۲, ۲۷۲۲–۲۷۲۸

[۷] Vladimir B.Bajic,Seng Hong Seah, Allen Chong,Guaglan Zhang, Judice L.Y.Koh and Vladimir

Brusic(٢٠٠٢), Dragon Promoter Finder:recognition of vertebrate RNA polymerase II Promoter,bioinformatics,vol ١٨,no. ١.١٩٨-١٩٩

[٨]http://www.genomatix.de/online_help/help_gems/PromoterInspector_help.html

[٩] Xiaomeng Li, Jia Zeng, and Hong Yan(٢٠٠٨), PCA-HPR: A principle component analysis model for human promoter recognition, Bioinformation by Biomedical Informatics Publishing Group.

[١٠] Pierre Baldi,Søren Brunak(٢٠٠١), Bioinformatics The Machine Learning Approach,second edition. Massachusetts Institute of Technology.

[١١] Ethem Alpaydin(٢٠١٠),Introduction to machine learning,second edition, The MIT Press Cambridge, Massachusetts London, England.

[١٢] Robi Polikar(٢٠٠٦),Ensemble Based Systems in Decision Making, ١٥٣١-٦٣٦X/٠٦/□٢٠,٠٠©٢٠٠٦ IEEE

[١٣] Artur Ferreira(٢٠٠٧), Survey on Boosting Algorithms for Supervised and Semi-supervised Learning, Instituto de Telecomunica¸c˜oes

[١٤]http://www.mathworks.com/help/stats/fitensemble.html

[١٥]http://www.mathworks.com/help/stats/classificationensembleclass.html