دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکدهٔ ریاضی و علوم‌کامپیوتر

گزارش ۷: مقایسه‌ی روش‌های مختلف طبقه‌بندی برای یک مسئله

نگارش
نیما حسینی دشت بیاض

استاد
دکتر مهدی قطعی

خرداد ۱۴۰۰

مقدمه

مسائل طبقه‌بندی از جمله مسائلی هستند که در حوزه‌ی یادگیری ماشین، الگوریتم‌های متنوعی برای آن‌ها طراحی شده است. در این مسائل، هدف الگوریتم پیش‌بینی کلاس هر ورودی است. به طور مثال، تشخیص اسپم بودن یا نبودن یک ایمیل، تشخیص ابتلا به یک بیماری، تشخیص حروف یک متن دست‌نویس و ... از نمونه‌های این نوع مسائل هستند.

در این تمرین عمل‌کرد چندین الگوریتم روی مسئله‌ی تشخیص اشیاء با سونار بررسی می‌شود.

شرح دیتاست

در این تمرین از دیتاست Sonar [1] استفاده شده است. این دیتاست شامل نمونه‌هایی از سیلندرهای فلزی و سنگ‌های استوانه‌ای است که برای هر کدام، قدرت امواج سونار در۶۰ زاویه‌ی متفاوت مشخص شده است. بنابراین مسئله دارای ۲ کلاس سنگ و فلز (Mine) است و هر نمونه دارای۶ ویژگی است.

تعداد کل نمونه‌های دیتاست ۲۰۸ عدد است. از این تعداد، ۴۰ مورد را برای تشکیل مجموعه‌ی تست انتخاب می‌کنیم. برای انتخاب این داده‌ها از روش Stratified Split استفاده می‌کنیم که تعادل میان دو کلاس را در داده‌های آموزش و تست حفظ می‌کند. همچنین برای الگوریتم‌های مبتنی بر Gradient Descent لازم است داده‌ها را استاندارد کنیم؛ به این معنا که ورودی‌ها اعداد میان ۱- و ۱ با میانگین صفر باشند. این کار باعث افزایش سرعت الگوریتم می‌شود. همچنین برای کلاس سنگ لیبل صفر و برای کلاس فلز لیبل ۱ را در نظر میگیریم.

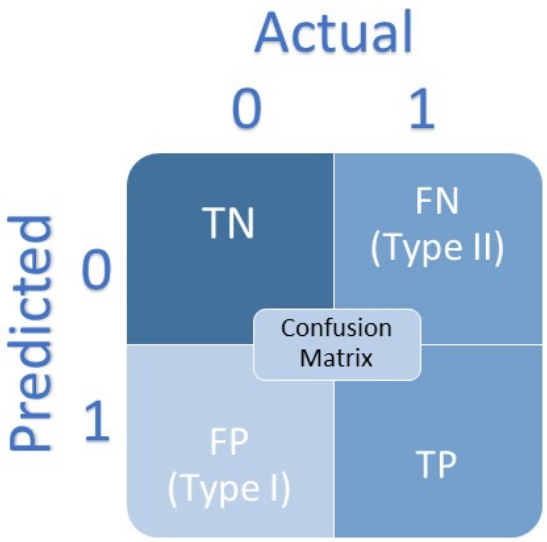
معیارهای ارزیابی

ماتریس درهم‌ریختگی[1]

ماتریس درهم‌ریختگی نشان‌دهنده‌ی تعداد پیش‌بینی‌های غلط و درست در هر کدام از کلاس‌هاست و به ما کمک می‌کند تا دید بهتری نسبت به عمل‌کرد یک مدل طبقه‌بندی دوتایی داشته باشیم. در این ماتریس چهار عدد قرار دارد که نشان‌دهنده‌ی تمام حالات کلاس پیش‌بینی شده هستند. به طور مثال، TN یا True Negatives نشان دهنده‌ی تعداد پیش‌بینی‌های درست از کلاس ۰ (منفی) است. به طور مشابه معیارهای True Positives, False Negatives و True Negatives هم تعریف می‌شوند. این به ما کمک می‌کند که

1 Confusion Matrix

عمل‌کرد را برای هر کلاس به‌طور جداگانه بررسی کنیم. در برخی مسائل مانند مسائب پزشکی و تشخیص بیماری، تشخیص مثبت بودن یک بیماری اهمیت بسیار بالایی دارد. به همین دلیل، در چنین مسئله‌ای پایین بودن تعداد False Negatives اهمیت بسیاز زیادی دارد؛ هرچند که ممکن است تعداد False Positive بالاتر باشد.[2



شکل ۱: ماتریس درهم‌ریختگی

بسیاری از معیارهای دیگر نیز براساس اطلاعات موجود در این ماتریس به دست می‌آید.

صحت یا Accuracy

این معیار تعداد پیش‌بینی‌های درست در همه‌ی کلاس‌ها را نشان می‌دهد.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

دقت یا Precision

این معیار نشان می‌دهد که چه درصدی از پیش‌بینی‌های یک کلاس درست بوده‌اند.

$$Precision = \frac{TP}{TP+FP}$$

فراخوانی یا Recall

فراخوانی یا Recall نشان می‌دهد که چه تعداد از کل داده‌هایی که یک کلاس به درستی در آن کلاس تشخیص داده شده‌اند.

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

F1 Score

این معیار یک میانگین هم‌ساز[۲] از دقت و فراخوانی است. میانگین همساز نوعی از میانگین است که به مقادیر کوچک‌تر اهمیت بیشتری می‌دهد. بنابرین مدلی امتیاز F1 بالایی خواهد داشت که هم دقت (درستی پیش‌بینی‌های کلاس مثبت) و هم فراخوانی (درصد داده‌های کلاس مثبت که درست تشخیص داده شده‌اند) بالایی داشته باشد.

$$F_1 = \frac{2}{\frac{1}{preision} + \frac{1}{recall}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

بررسی روش‌های یادگیری [3]

Logistic Regression

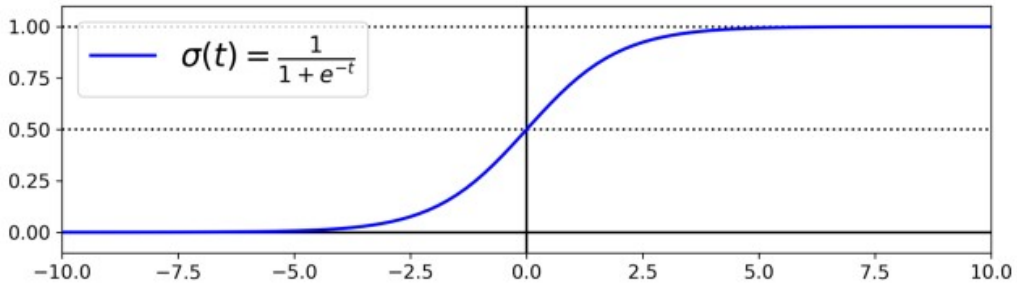
در این الگوریتم مشابه روش Linear Regression، یک مجموع وزن‌دار (ترکیب خطی) از ویژگی‌های مختلف محاسبه می‌شود نتیجه‌ی آن به تابع Logistic داده می‌شود تا احتمال تعلق به کلاس ۱ محاسبه شود. در صورتی که این احتمال بیش از ۰.۵ باشد، کلاس ۱ و در غیر این صورت، کلاس صفر پیش‌بینی می‌شود. تابع Logistic که با σ نمایش داده می‌شود همان تابع Sigmoid است که در پایین نشان داده شده است.

$$y = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$\sigma(y) = \frac{1}{1 + \mathbf{exp}(-y)}$$

مقدار $\sigma(y)$ در فرمول بالا احتمال تعلق $\mathbf{x}$ به کلاس ۱ را نشان می‌دهد.

2 Harmonic Mean



شکل ۲: نمودار تابع Logistic

پیاده‌سازی

برای استفاده از این روش برای یادگیری دیتاست، از کتاب‌خانه‌ی Scikit-Learn و کلاس LogistRegression استفاده می‌کنیم. این کلاس برای به دست آوردن وزن‌های مدل از روش Normal Equation استفاده می‌کند که مبتنی بر SVD است.

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

همچنین برای جلوگیری از overfit شدن داده‌ها از پنالتی L2 استفاده می‌کنیم که عبارت $\alpha\frac{1}{2}\Sigma_{i=1}^{n}\theta_i^2$ را به تابع خطا اضافه می‌کند.

```
log_reg = LogisticRegression(penalty='l2')
log_reg = log_reg.fit(X_train_std, y_train)
```

ارزیابی

معیارهایی که در قسمت قبل توضیح داده شد را برای داده‌های تست بررسی می‌کنیم. برای محاسبه‌ی این معیارها از توابع confusion_matrix و classification_report که توسط Scikit-Learn ارائه می‌شود استفاده می‌کنیم.

```
print_confusion(y_test, y_preds_reg)
```

```
11 6
8 15
```

```
print(classification_report(y_test, y_preds_reg))
```

	precision	recall	f1-score	support
0	0.65	0.58	0.61	19
1	0.65	0.71	0.68	21
accuracy			0.65	40
macro avg	0.65	0.65	0.65	40
weighted avg	0.65	0.65	0.65	40

شکل ۳: معیارهای ارزیابی برای روش Logistic Regression

در تصویر فوق، ماتریس درهم‌ریختگی نشان داده شده است. ۶ مورد از ۲۱ نمونه‌ی کلاس Mine به اشتباه سنگ تشخیص داده شده است و به طور مشابه ۸ مورد از ۱۹ نمونه سنگ هم در کلاس Mine قرار گرفته‌اند. سایر معیارهای ارزیابی برای هر کلاس نیز مشخص شده است.

استفاده از Polynomial Features

در روش Logistic Regression فقط یک ترکیب خطی از ویژگی‌ها محاسبه می‌شود. به همین دلیل، روابط میان ویژگی‌ها ممکن است مشخص نشود. به طور مثال، پیدا کردن خطی که تابع x^2 را فیت کند ممکن نیست. با اضافه کردن درجات بالاتر ویژگی‌ها و حاصل ضرب آن‌ها، می‌توان الگوهای پیچیده‌تر که از روابط بین ویژگی‌ها ایجاد می‌شوند را پوشش داد. با افزودن Polynomial Features درجه ۲ به ویژگی‌ها، می‌توان feature های جدیدی برای داده‌ها در نظر گرفت که به صورت $x_i x_j$ ها به ازای هر x_i و x_j در ویژگی‌های اصلی تعریف می‌شوند. حال می‌توان روش Logistic Regression را با داده‌های جدید استفاده کرد.

برای انجام این کار از پکیج PolynomialFeatures در Scikit-Learn استفاده می‌شود. در تصویر زیر ارزیابی این روش مشخص شده است.

```
print_confusion(y_test, y_preds_poly_reg)
```

```
18 11
1 10
```

```
print(classification_report(y_test, y_preds_poly_reg))
```

```
              precision    recall  f1-score   support

           0       0.62      0.95      0.75        19
           1       0.91      0.48      0.62        21

    accuracy                           0.70        40
   macro avg       0.76      0.71      0.69        40
weighted avg       0.77      0.70      0.68        40
```

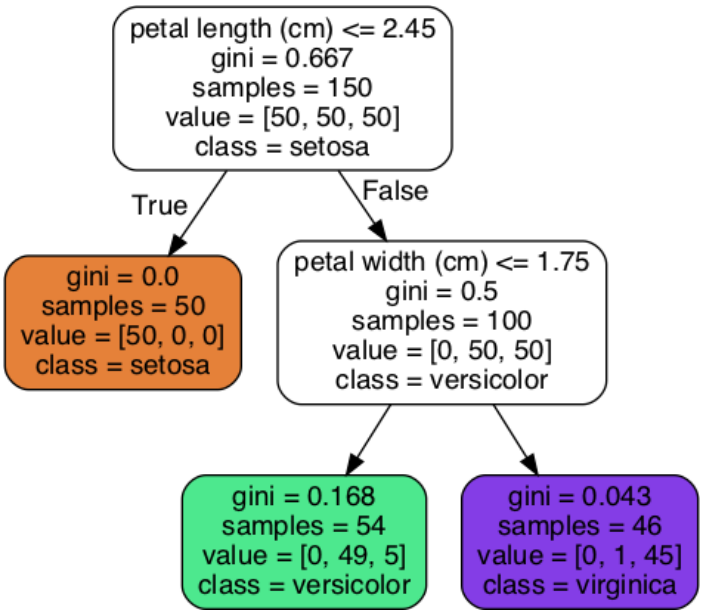
شکل ۴: ارزیابی روش Logistic Regression با ویژگی‌های درجه‌ی دوم

در این روش تعداد زیادی از نمونه‌های Mine در کلاس سنگ قرار گرفته‌اند. به همین دلیل دقت تشخیص سنگ‌ها هم بالا رفته است.

درخت تصمیم[3]

در روش درخت تصمیم، یک درخت تشکیل می‌شود که در هر نود میانی، یک شرط بر روی ویژگی‌های داده‌ها تعریف می‌شود. اگر این شرط برقرار باشد به فرزند سمت چپ آن می‌رویم و در غیر این صورت فرزند سمت راست را بررسی می‌کنیم. با رسیدن به برگ‌ها، شرطی باقی نمی‌ماند و برای هر برگ یک کلاس در نظر گرفته می‌شود و داده‌هایی که داخل آن برگ قرار می‌گیرند در آن کلاس طبقه‌بندی می‌شوند.

هنگام یادگیری، این الگوریتم سعی می‌کند شرط‌ها را به گونه‌ای تنظیم کند که impurity داده‌های داخل نودهای میانی و برگ‌ها حداقل شود؛ به این معنا که تا حد امکان داده‌ها از یک کلاس باشند.



شکل ۵: یک درخت تصمیم برای دیتاست Iris

پیاده‌سازی

برای استفاده از این الگوریتم از کلاس DecisionTreeClassifier در پکیج Scikit-Learn استفاده می‌کنیم. پارامترهای زیادی برای محدود کردن درخت‌های تصمیم وجود دارد که برای جلوگیری از overfit شدن کاربرد دارد؛ مانند حداقل تعداد نمونه‌هایی که باید داخل یک برگ باشند، حداقل نمونه‌ی لازم برای ایجاد یک split، حداکثر عمق درخت و... . پس از بررسی پارامترهای مختلف، درخت بدون داشتن محدودیت بهترین نتیجه را برای داده‌های تست داشت.

```
dec_tree = DecisionTreeClassifier()
dec_tree = dec_tree.fit(X_train, y_train)
```

3 Decision Tree

ارزیابی

```
print_confusion(y_test, y_preds_tree)
```

```
13 7
6 14
```

```
print(classification_report(y_test, y_preds_tree))
```

	precision	recall	f1-score	support
0	0.65	0.68	0.67	19
1	0.70	0.67	0.68	21
accuracy			0.68	40
macro avg	0.68	0.68	0.67	40
weighted avg	0.68	0.68	0.68	40

شکل ۶: ارزیابی روش درخت تصمیم

همان‌طور که در تصویر بالا مشخص است، این روش عمل‌کرد بهتری در معیارهای مختلف نسبت به روش‌های قبلی داشته است.

جنگل تصمیم تصادفی[۴]

جنگل‌های تصمیم یکی از انواع روش‌های جمعی یا Ensemble هستند که قدرت بسیار بالایی دارند. به طور کلی در این روش‌ها به جای استفاده از یک مدل، از چندین مدل مختلف استفاده می‌شود و تصمیم نهایی برآیندی از تصمیم هر کدام از این مدل‌ها خواهد بود. این روش‌ها قدرت این را دارند که نقاط ضعف و مشکلات یک‌دیگر را پوشش دهند؛ مخصوصاً زمانی که هر مدل خطای متفاوتی از دیگری داشته باشد.

جنگل‌های تصمیم از اجماع چندین درخت تصمیم ایجاد می‌شود و در هر درخت فقط زیر مجموعه‌ای تصادفی از ویژگی‌های داده‌های ورودی برای تصمیم‌گیری استفاده می‌شود. همچنین با افزودن محدودیت‌هایی مختلف به هر کدام از درخت‌ها، از overfit شدن تک درخت‌ها جلوگیری می‌شود و با داشتن تعداد زیادی درخت می‌توان تصمیمی گرفت که بیشترین احتمال را دارد.

پیاده‌سازی

برای باستفاده از این روش از کلاس RandomForestClassifier استفاده می‌کنیم. تعداد درخت‌ها را برابر ۱۰۰۰ درخت قرار می‌دهیم و حداقل تعداد نمونه‌ی لازم برای ایجاد split در هر درخت را برابر ۵ نمونه قرار

4 Random Forest

می‌دهیم. با این محدودیت‌ها، مدل عمل‌کرد بهتری را در مجموعه‌ی تست نسبت به حالت بدون محدودیت دارد.

```
forest = RandomForestClassifier(n_estimators=1000, min_samples_split=5)
forest = forest.fit(X_train, y_train)
```

ارزیابی

```
print_confusion(y_test, y_preds_forest)
```

```
14 3
5 18
```

```
print(classification_report(y_test, y_preds_forest))
```

	precision	recall	f1-score	support
0	0.82	0.74	0.78	19
1	0.78	0.86	0.82	21
accuracy			0.80	40
macro avg	0.80	0.80	0.80	40
weighted avg	0.80	0.80	0.80	40

شکل ۷: ارزیابی روش جنگل تصمیم

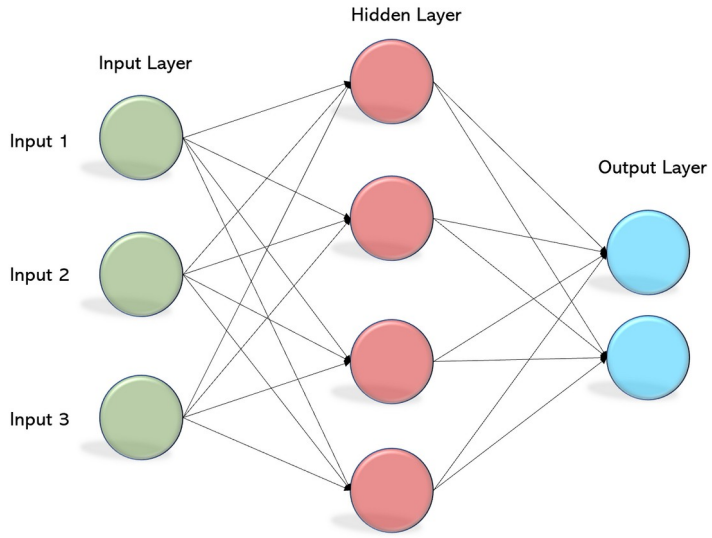
همان‌طور که انتظار داریم، این روش عملکرد بسیار بهتری نسبت به روش‌های قبلی و درخت تصمیم دارد و دارای دقت ۸۰ درصد است.

شبکه‌ی عصبی چندلایه[5]

شبکه‌های MLP از چند لایه نورون تشکیل شده‌اند که هر نورون به تمام نورون‌های لایه‌ی بعدی متصل است. هر نورون یک ترکیب خطی از تمام ورودی‌هایش را محاسبه می‌کند و سپس با استفاده از یک تابع فعال‌ساز امکان یادگیری الگوهای غیرخطی را ایجاد می‌کند.

شبکه‌های عصبی به دلیل توانایی تشخیص الگوهای پیچیده در داده‌ها قدرت بالایی دارند و مورد توجه قرار گرفته‌اند؛ اما مشکل اصلی این مدل‌ها نیاز آن‌ها به تعداد زیادی داده برای یادگیری است. در این مسأله با توجه به اینکه تنها حدود ۱۵۰ داده‌ی یادگیری در دسترس است (بدون احتساب داده‌های validation)، نمی‌توان انتظار زیادی از شبکه‌ی عصبی داشت. با این حال بازم هم عمل‌کردی هم‌سطح با روش‌های قبل قابل انتظار است.

5 Multi Layer Perceptron



شکل ۸: نمونه‌ای از یک شبکه‌ی MLP با یک لایه‌ی مخفی

پیاده‌سازی

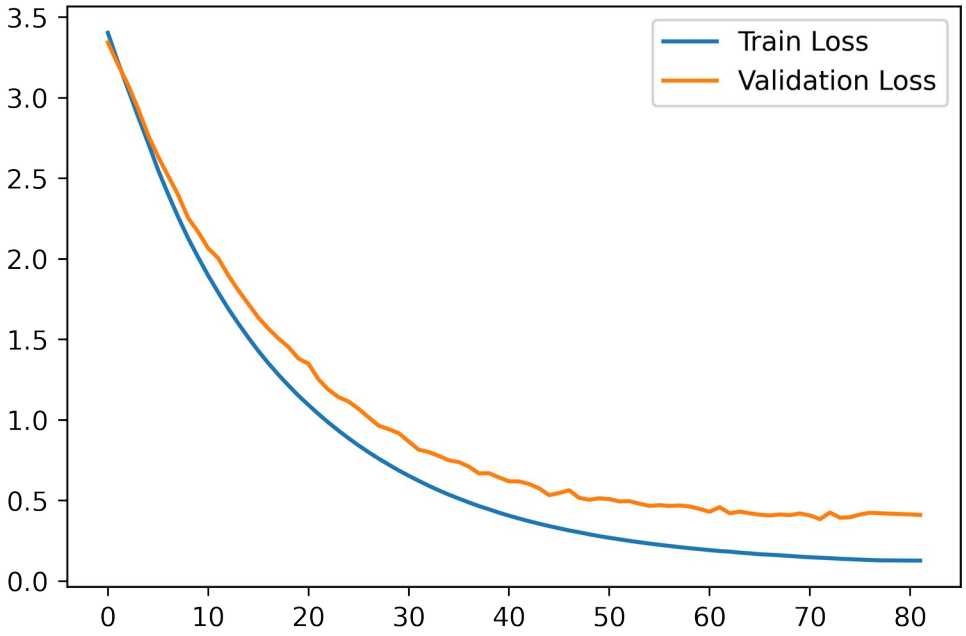
برای پیاده‌سازی شبکه‌ی عصبی از کتاب‌خانه‌های Keras و TensorFlow استفاده می‌کنیم. در مدل پیاده شده، سه لایه‌ی مخفی قرار دارد که هر کدام دارای ۱۰۰ نورون هستند و تابع فعال‌ساز همه‌ی آن‌ها relu است. همچنین همه‌ی لایه‌ها دارای L2 Regularization هستند تا مانع overfit شدن شود. لایه‌ی آخر شبکه دارای یک نورون با تابع فعال‌ساز sigmoid است.

```
deep_model = keras.models.Sequential([
    keras.layers.InputLayer(input_shape=X_train_std.shape[1]),
    keras.layers.Dense(100, activation="relu",  kernel_regularizer='l2'),
    keras.layers.Dense(100, activation="relu",  kernel_regularizer='l2'),
    keras.layers.Dense(100, activation="relu", kernel_regularizer='l2'),
    keras.layers.Dense(1, activation="sigmoid")
])
```

شکل ۹: مدل شبکه‌ی MLP

مدل از الگوریتم Nadam برای یادگیری استفاده می‌کند که سرعت بیشتری از Gradient Descent و Adam دارد. همچنین از دو روش Early Stopping برای توقف الگوریتم در صورت عدم بهبود و Reduce on Plateau برای کاهش Learning Rate هم استفاده شده است. در این مدل برای کنترل یادگیری از دیتای validation هم استفاده شده که ۱۵ درصد دیتای یادگیری است.

یادگیری در epoch ۸۱ توسط Early Stopping متوقف می‌شود. خطا و دقت دیتای ولیدیشن به ۹۲ درصد و ۰.۳۴۹ می‌رسد



شکل ۱۰: خطای یادگیری و validation برای شبکه

ارزیابی

```
print_confusion(y_test, y_preds_deep)
```

```
14 3
5 18
```

```
print(classification_report(y_test, y_preds_deep))
```

	precision	recall	f1-score	support
0	0.82	0.74	0.78	19
1	0.78	0.86	0.82	21
accuracy			0.80	40
macro avg	0.80	0.80	0.80	40
weighted avg	0.80	0.80	0.80	40

شکل ۱۱: ارزیابی شبکه‌ی MLP

شبکه دارای دقت ۸۰ درصدی روی داده‌های تست است که مشابه مدل جنگل تصمیم است و از سایر الگوریتم‌های بررسی شده دقت بهتری دارد. تفاوت عمل‌کرد این روش با جنگل نوع خطاهاست. در جنگل تعداد FN برابر ۵ و FP برابر ۳ است که برعکس شبکه است.

کد پروژه

کد کامل پروژه به‌همراه خروجی‌های موجود در این گزارش در آدرس گیت‌هاب زیر قرار دارد.

https://github.com/nimahsn/AI-Course-Projects/tree/main/report07-ML_benchmark

جمع‌بندی

در این گزارش چهار الگوریتم متفاوت بررسی شد که در بین آن‌ها جنگل تصمیم و شبکه‌ی عصبی بهترین عمل‌کرد را داشتند. همچنین با استفاده از روش‌هایی مانند Cross Validation و یافتن پارامترهای بهتر می‌توان این روش‌ها را بهبود داد. علاوه بر آن، می‌توان با استفاده از مدل‌های شبکه‌ی عصبی و جنگل تصمیم که برای این گزارش پیاده شدند، یک مدل Ensemble جدید نیز تشکیل داد؛ چرا که در نوع خطای شبکه و جنگل تفاوت وجود داشت که یک مدل Ensemble می‌تواند آن را بهبود دهد.

منابع

1 http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)
2 https://stanford.edu/~shervine/l/fa/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks
3 Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.