به نام خدا

عنوان درس:

Machine learning

گزارش تمرین 2:

ID3 and random forest algorithms

تاریخ تحویل:

03/02/1401

فاطمه حسن زاده

40032217

امیرحسن امیرماهانی

40032722

مقدمه:

در این پروژه،در دو بخش A و B ،الگوریتم درخت تصمیم ID3 و الگوریتم Random forest را پیاده سازی می کنیم.

فرض می‌کنیم که بانکی می‌خواهد فرآیند واجد شرایط بودن وام را بر اساس جزئیات مشتری که در هنگام پر کردن فرم درخواست آنلاین ارائه شده ، پیش‌بینی(predict) کند.

در پارت A با استفاده از الگوریتم ID3 پیش بینی مدنظر را انجام دهیم و در اخر دقت(accuracy) به دست آمده را گزارش می کنیم.

سیستم تشخیص نفوذ (IDS) یک دستگاه یا نرم افزار کاربردی است که یک شبکه یا سیستم را برای فعالیت های مخرب یا نقض خط مشی نظارت می کند. برای افزایش امنیت شبکه می توان از الگوریتم های یادگیری ماشین برای شناسایی و جلوگیری از حملات شبکه استفاده کرد. طبق تحقیقات انجام شده، یکی از محبوب ترین الگوریتم های یادگیری ماشین در زمینه سیستم های تشخیص نفوذ، الگوریتم جنگل تصادفی(Random forest) است.

در پارت B با استفاده از دیتاست NSL-KDD و الگوریتم Random forest ، مدلی را برای شناسایی اکشن های مخرب یا عادی پیاده سازی می¬کنیم، سپس دقت به دست آمده را گزارش میکنیم.

پارت A :

الگوریتم ID3

الگوریتم ID3، درخت های تصمیم را با ساختن آنها از بالا به پایین یاد می گیرد(learning).

در ساخت درخت تصمیم یک سوال مهم است و آن این است که " کدام مشخصه باید برای گره فعلی انتخاب شود؟". این سوال در ID3 با اندازه گیری معیار information gain حل می شود.

بهترین مشخصه یعنی مشخصه ای که بیشترین gain را دارد، در ریشه درخت قرار می¬گیرد.

Information gain :

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Entropy :

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

با توجه به اینکه برخی از این ویژگی‌ها

int.rate ،installment ،log.annual.inc ،dti ،fico،days.with.cr.line ،revol.util ، revol.bal

داده‌های پیوسته هستند ، باید با استفاده از الفاکات(cutpoint) مناسب آن‌ها را گسسته کنیم.

که با 3 روش مختلف می¬توان این تقسیم بندی داده ها را انجام داد:

مینیم انتروپی : یعنی الفاکاتی که بیشترین کاهش انتروپی را ایجاد کند.

ماکسیمم gain : یعنی الفاکاتی که بیشترین افزایش gain را به همراه داشته باشد.

تقسیم کردن به 5 دسته مساوی

در ادامه به توضیح پیاده سازی قسمت های بالا می¬پردازیم:

تابع splitter(df,A) :

این تابع دیتاست مربوطه و ویژگی های پیوسته را به عنوان ورودی دریافت کرده و برای هر ویژگی الفا کاتی را پیدا می کند ک مینیمم انتروپی را داشته باشد، همچنین الفاکاتی را پیدا می کند که ماکسیمم gain را داشته باشد.سپس داده ها را بر اساس این الفاکات تقسیم میکند.

تابع normalize(df) :

این تابع دیتاست را به عنوان ورودی دریافت می¬کند و با توجه به تقسیم بندی داده های پیوسته که بر اساس 3 روش مذکور انجام می شود، آن دیتاست را نرمال سازی می¬کند.

و به مقادیر هر دسته Lable مناسب را با استفاده از تابع labler(value,labels) اختصاص می¬دهد.

تابع labler(value,labels) :

این تابع مقادیر ویژگی ها و لیبل ها را دریافت کرده و به هر مقدار ،لیبل مناسب آن را اختصاص می¬دهد.

کلاس ID3 :

در این کلاس به طور کلی entropy و gain را محاسبه می¬کند و با ایتفاده از آن ها درخت تصمیم را تشکیل می¬دهد و در اخر پیش بینی مدنظر را انجام می¬دهد.

تابع entropy(self, feature_column) :

این تابع مقادیر یک ویژگی را به عنوان ورودی دریافت کرده و انتروپی آن را با توجه به فرمولی که در بالا ذکر کردیم، محاسبه می¬کند.

تابع information_gain(self, data, feature_name, target_name) :

این تابع دیتاست و نام ویژگی و نام تارگت مدنظر را دریافت کرده و با توجه ب فرمولی که برای gain ذکر شد و با کمک تابع انتروپی، gain را محاسبه میکند.

تابع decision_tree(data, basic_data, features_name, target, parent_node_class) :

این تابع به صورت کلی درخت تصمیم را تشکیل می¬هد.

به این صورت که زیرمجموعه ای از دیتاست که عملیات روی آن انجام میشود(data)، دیتاست اصلی، نام ویژگی ها، نام ویژگی تارگت و parent_node_class را دریافت کرده و با استفاده از تابع unique() مقادیر ویژگی تارگت در زیرمجموعه data را کلاس بندی میکندو نام همه کلاس ها را در متغیری به نام unique_classes می¬ریزد.

اگر تعداد کلاس ها برابر یک بود یعنی دیتا pure است و مقدار آن کلاس را برمی¬گرداند.

اگر زیر مجموعه مدنظر خالی بود، کلاسی که بیشترین مقدار را در دیتاست اصلی دارد، برمی¬گرداند.

اگر تعداد ویژگی ها برابر صفر بود آنگاه parent_node_class را برمی¬گرداند.

در اخر اگر هیچ کدام از شرط های بالا برقرار نبود آنگاه به صورت زیر عمل می¬کند:

در ابتدا کلاسی که بیشترین مقدار را دارد در متغیر parent_node_class می¬ریزد.

سپس در یک حلقه for برای هر ویژگی gain آن با صدا زدن تابع information_gain را محاسبه می¬کند. ویژگی ای که بیشترین gain را دارد در متغیر best_feature نگهداری می¬کند و آن را در ریشه شاخه درخت قرار می¬دهد و از مجموعه ویژگی ها حذف میکند و مقادیر این ویژگی را با استفاده از تابع unique() کلاس بندی کرده در متغیر parent_values می¬ریزد.

و در آخر در یک حلقه for به ازای هر مقدار موجود در parent_values تابع decision_tree را به صورت بازگشتی صدا می¬زند. و به صورت عمقی کل درخت را تشکیل می¬دهد.

تابع make_prediction(self, sample, tree, default=1) :

این تابع نمونه ها و درخت رو دریافت کرده و هر نمونه را در درخت map می¬کند.

تابع predict(self, input) :

این تابع بر روی نمونه دریافتی را با کمک تابع make_prediction() پیش بینی را انجام می¬دهد.

تابع fit(self, input, output) :

این تابع برای input و output دریافتی تابع درخت تصمیم را صدا میزند. Input بیانگر نام ویژگی ها و output بیانگر نام ویژگی تارگت است.

Main :

در قسمت main ،ابتدا دیتاست بانک را نرمال سازی می¬کنیم و مقادیر ویژگی "credit.policy" را که ویژگی تارگت ما هست را در متغیر X می¬ریزیم. همچنین مقادیر این ویژگی را از دیتاست بانک حذف کرده و در متغیر Y نگهداری می¬کنیم.

با استفاده از تابع train_test_split(X,Y) از کتابخانه sklearn.model_selection، 20 درصد داده ها را برای تست و 80 درصد را برای train تقسیم میکنیم و نتیجه را در 4 متغیر X_train, X_test, y_train, y_test نگه میداریم.

از کلاس ID3 یک شی ایجاد کرده و تابع fit آن را با ورودی های X_train و y_train صدا میزنیم تا بر روی دیتاست مدنظر train را انجام دهد و درخت تصمیم را تشکیل دهد.

در اخر تابع predict() را با ورودی X_test صدا میزنیم و نتیجه را در متغیر y_pred میریزیم و بخش test را انجام میدهیم.

آنگاه دقت مورد نظر را با استفاده از تابع accuracy_score(y_test, y_pred) از کتابخانه sklearn.metrics محاسبه میکنیم.

نتایج :

اگر داده های پیوسته با استفاده از روش مینیمم انتروپی تقسیم کنیم و دیتاست را بر این اساس نرمال کنیم آنگاه دقت به دست آمده برابر 87 درصد است.

اگر داده های پیوسته با استفاده از روش ماکسیمم gain تقسیم کنیم و دیتاست را بر این اساس نرمال کنیم آنگاه دقت به دست آمده برابر 86 درصد است.

اگر مقادیر هر ویژگی را به 5 دسته مساوی تقسیم کنیم و دیتاست را بر این اساس نرمال کنیم آنگاه دقت به دست آمده برابر 91 درصد است.

با توجه به اینکه یکی از ویژگی هایی که مقادیر پیوسته دارد (revol.util) و در متن هموورک در قسمت ویژگی های پیوسته به آن اشاره نشده ما بدون در نظر گرفتن پیوسته بودن مقادیر این ویژگی نیز دقت را محاسبه کردیم و به دقت 85٪ رسیدیم.

در ادامه بخشی از مقادیر محاسبه شده برای قسمت 1 و 2 را مشاهده می¬کنیم:

@int.rate Initial Entropy = 1.98899033021754

@int.rate, alpha =2 : Gain is -1.1452652488248105

@int.rate, alpha =3 : Gain is -10.573714431524339

@int.rate, alpha =4 : Gain is -24.012088924246445

@int.rate, alpha =5 : Gain is -42.07580089548233

int.rate max gain alpha 2

@int.rate Initial Entropy = 1.98899033021754

@int.rate, alpha =2 : Entropy is -34443.17035458937

@int.rate, alpha =3 : Entropy is -73344.44963459326

@int.rate, alpha =4 : Entropy is -63515.0969750911

@int.rate, alpha =5 : Entropy is -70014.45158213607

int.rate min entropy alpha is 2

مابقی نتایج به پیوست تقدیم میگردد.

پارت B :

الگوریتم Random forest

الگوریتم Random forest ، یک روش یادگیری ensemble برای classification ، رگرسیون و سایر وظایف است که با ساختن تعداد زیادی درخت تصمیم در زمان train کردن عمل می‌کند.

برای کارهای classification ، خروجی جنگل تصادفی کلاسی است که توسط اکثر درختان انتخاب شده است.

دراین بخش ابتدا فایل های مربوط به دیتاست NSL-KDD را می¬خوانیم و کامنت ها را مشخص میکنیم و هدر عددی برای ستون ها در نظر میگیریم.

سپس به کمک لایبرری sklearn.ensemble و کلاس RandomForestClassifier داده ها را دسته بندی می¬کنیم و با استفاده از تابع fit() و predict() ، عمل train و test را انجام می¬دهیم و دقت را محاسبه می-کنیم.

سپس با کمک کلاس AdaBoostClassifier در کتابخانه مذکور دسته بندی داده ها را انجام می¬دهیم و مانند قبل عمل می¬کنیم.

نتایج به دست آمده را در زیر مشاهده می¬کنیم که همانطور که از اعداد به دست امده مشخص است در مود boosting عملکرد بهتری داشته است

نتایج :

----------RandomForestClassifier----------

accuracy: 0.7404187366926899

percision: 0.627772995165883

f1: 0.7641463807834918

recall: 0.9762125424776027

----------AdaBoostClassifier----------

accuracy: 0.7824698367636622

percision: 0.6709581051283875

f1: 0.793689524610854

recall: 0.9713726701678509