پیاده سازی درخت تصمیم با استفاده از الگوریتم آنتروپی

1. تابع entropy : این تابع برای محاسبه آنتروپی (entropy) یک متغیر هدف (target variable)استفاده می‌شود. ابتدا تعداد هر کلاس در داده‌ها شمرده شده و سپس احتمالات آن‌ها محاسبه می‌شود. سپس مقدار entropy محاسبه میشود.

2 .تابع information_gain : این تابع برای محاسبه information gain برای یک ویژگی و مقدار تقسیم (split value) خاص استفاده می‌شود. ابتدا ماسک‌هایی برای داده‌ها ایجاد می‌شود که بر اساس مقدار تقسیم، داده‌ها را به دو دسته تقسیم می‌کند. سپس entropy برای هر دسته محاسبه می‌شود و با وزن متناسب با تعداد داده‌های هر دسته، information gainمحاسبه می‌شود.

3 .تابع best_split :تابع بهترین ویژگی و مقدار تقسیم را بر اساس مقدار بیشینه information gainپیدا می‌کند. برای هر ویژگی، تمام مقادیر یکتا را در نظر می‌گیرد و برای هر مقدار تقسیم ویژگی، information gainرا محاسبه می‌کند و با مقدار بیشترین information gainمقایسه کرده و بهترین ویژگی و مقدار تقسیم را ذخیره می‌کند.

4 .تابع build_tre: :این تابع به صورت بازگشتی درخت تصمیم را ساخت می‌کند. ابتدا بررسی می‌شود که آیا شرایط متوقف شدن برای ساخت درخت برقرار است یعنی عمق ماکسیمم درخت رسیده است یا تمام داده‌ها یکتا هستند. اگر این شرایط برقرار باشد، یک برگ درخت با کلاس پرتکرارترین داده‌ها ساخته می‌شود. در غیر این صورت، بهترین ویژگی و مقدار تقسیم را با استفاده از تابع best_split پیدا کرده و درخت را با تقسیم داده‌ها بر اساس این ویژگی و مقدار ساخته و به صورت بازگشتی بر روی هر زیردرخت فراخوانی می‌شود.

5 .تابع fit این تابع برای ساخت مدل درخت تصمیم، متد `fit` از کلاس `DecisionTree` فراخوانی می‌شود. این متد به عنوان ورودی داده‌های آموزشی را می‌گیرد و با استفاده از متدهای دیگر کلاس، مدل درخت تصمیم را ساخته و آموزش می‌دهد. در این متد، ابتدا تمام ویژگی‌ها و مقادیر یکتای آن‌ها در داده‌های آموزشی بررسی می‌شوند و بهترین ویژگی و مقدار تقسیم را با استفاده از تابع `best_split` پیدا می‌کند. سپس درخت تصمیم با تقسیم داده‌ها بر اساس این ویژگی و مقدار ساخته می‌شود. این اقدام به صورت بازگشتی بر روی هر زیردرخت انجام می‌شود تا در نهایت درخت کامل شود.

بعد از ساخت مدل درخت تصمیم، می‌توان از آن برای پیش‌بینی برچسب‌های کلاس جدید استفاده کرد. متد `predict_instance` برای پیش‌بینی برچسب یک نمونه جدید استفاده می‌شود. این متد نمونه را به عنوان ورودی می‌گیرد و با استفاده از درخت تصمیم، به صورت بازگشتی بر روی زیردرخت‌ها حرکت کرده و برچسب نهایی را بازگردانده می‌کند.

علاوه بر این، متد `predict` نیز برای پیش‌بینی برچسب‌های کلاس برای یک مجموعه نمونه استفاده می‌شود. این متد مجموعه نمونه‌ها را به عنوان ورودی می‌گیرد و برای هر نمونه، متد `predict_instance` را فراخوانی می‌کند تا برچسب نهایی را برای آن نمونه پیش‌بینی کند. نتیجه پیش‌بینی برچسب‌ها به صورت یک آرایه بازگردانده می‌شود.

علاوه بر پیش‌بینی برچسب‌ها، می‌توان با استفاده از تابع `print_tree` ساختار درخت تصمیم را نمایش داد. این تابع به صورت بازگشتی بر روی درخت حرکت کرده و برای هر گره، ویژگی و مقدار تقسیم را نمایش می‌دهد.

دستور `data = pd.read_csv("onlinefraud.csv")[['step','nameOrig','oldbalanceOrg','newbalanceOrig','nameDest','amount', 'isFraud', 'oldbalanceDest', 'newbalanceDest']].head(2000).dropna()` داده‌ها را از فایل CSV می‌خواند و سپس ستون‌های مشخصی از آن‌ها را انتخاب می‌کند. سپس تنها اولین 2000 ردیف از داده‌ها را نگه می‌دارد و ردیف‌هایی که دارای مقادیر نامعتبر (NA) هستند را حذف می‌کند.

سپس با استفاده از `data.groupby('nameOrig')['isFraud'].count()` تعداد رکوردهای هر نام مبدا (nameOrig) را محاسبه و نمایش می‌دهد.

سپس با استفاده از `data.groupby('nameDest')['nameDest'].count()` تعداد رکوردهای هر نام مقصد (nameDest) را محاسبه و نمایش می‌دهد.

دستورات `label_encoder.fit_transform(data['nameOrig'])` و `label_encoder.fit_transform(data['nameDest'])` برای تبدیل ستون‌های "nameOrig" و "nameDest" به اعداد صحیح استفاده می‌شود.

در ادامه، مدلی برای پیش‌بینی احتمال تقلب (isFraud) بر اساس ویژگی‌های داده‌ها ایجاد می‌شود. ابتدا داده‌ها به دو بخش آموزشی و تست تقسیم می‌شوند با استفاده از `train_test_split`، سپس یک مدل درخت تصمیم (Decision Tree) با استفاده از `DecisionTree(max_depth=5)`آموزش داده می‌شود.

سپس با استفاده از `decision_tree.print_tree()` ساختار درخت تصمیم نمایش داده می‌شود.

سپس با استفاده از مدل آموزش دیده، پیش‌بینی بر روی داده‌های تست انجام می‌شود و دقت پیش‌بینی با استفاده از `accuracy_score` محاسبه و نمایش داده می‌شود.

سپس یک مدل دیگر از درخت تصمیم با استفاده از `DecisionTreeClassifier(max_depth=3, min_samples_split=100)`آموزش داده می‌شود.

سپس با استفاده از اعتبارسنجی متقابل (cross-validation) با استفاده از `cross_val_score`، عملکرد مدل روی کمجموعه‌داده در خروجیچاپ می‌شود. نتایج عملکرد مدل در هر بار تقسیم داده به 5 بخش و اجرای اعتبارسنجی متقابل نمایش داده می‌شود.

سپس ویژگی‌ها (X) و برچسب‌ها (y) از داده‌ها استخراج می‌شوند و به دو بخش آموزشی و تست تقسیم می‌شوند.

در نهایت، یک مدل دیگر از درخت تصمیم (Decision Tree Classifier) با استفاده از `DecisionTreeClassifier(max_depth=3, min_samples_split=100)`آموزش داده می‌شود.

در انتها، عملکرد مدل با استفاده از اعتبارسنجی متقابل نمایش داده می‌شود.

تابع groupby برای تقسیم داده ها به گروه ها بر اساس ستون(های) مشخص شده استفاده می شود و سپس یک تابع تجمیع (مانند تعداد یا میانگین) در ستون "isFraud" در هر گروه اعمال می شود. نتایج بینش هایی را در مورد توزیع یا آمار خلاصه متغیر "isFraud"بر اساس ستون(های) گروه بندی شده ارائه می دهد

با انجام رمزگذاری برچسب، متغیرهای دسته بندی مانند 'nameOrig' و 'nameDest' به نمایش های عددی تبدیل می شوند. این به مدل درخت تصمیم اجازه می دهد تا با این متغیرها به عنوان ویژگی های ورودی کار کند، زیرا درخت های تصمیم معمولاً به ورودی عددی نیاز دارند. مقادیر رمزگذاری شده روابط زیربنایی بین دسته‌ها را حفظ می‌کنند و در عین حال یک نمایش عددی ارائه می‌دهند که می‌تواند توسط مدل پردازش شود

داده‌های ورودی به دو بخش آموزش (X_train و y_train) و آزمون (X_test و y_test) تقسیم می‌شوند. با استفاده از بخش آموزش، می‌توانید مدل درخت تصمیم را آموزش دهید و با استفاده از بخش آزمون، عملکرد مدل را بر روی داده‌های جدید ارزیابی کنید. این به شما کمک می‌کند تا به اندازه‌گیری قدرت تعمیم‌پذیری مدل برای داده‌های ناشناخته بپردازید.

y_pred = decision_tree.predict(X_test: این خط کد برای پیش‌بینی برچسب کلاس بر روی داده‌های آزمون (X_test) با استفاده از مدل درخت تصمیم (decision_tree) استفاده می‌شود. تابع predict مدل را با داده‌های آزمون ورودی تغذیه کرده و برچسب کلاس پیش‌بینی شده را برمی‌گرداند. این برچسب‌ها در y_pred ذخیره می‌شوند.

cv_scores = cross_val_score(decision_tree, X, y, cv=5)
کد بالا برای ارزیابی عملکرد مدل درخت تصمیم با استفاده از اعتبارسنجی متقاطع (Cross-Validation) استفاده می‌شود.

پیاده سازی درخت تصمیم با استفاده از الگوریتم gini index

__init__(self, Y, X, min_samples_split=None, max_depth=None, depth=None, node_type=None, rule=None): متد سازنده کلاس است که پارامترهای مختلف را دریافت می‌کند و ویژگی‌های مربوط به یک نود را مقداردهی اولیه می‌کند.

GINI_impurity(y1_count, y2_count): یک متد استاتیک برای محاسبه میزان خلوص GINI بر اساس تعداد دو کلاس مختلف است.

ma(x, window): یک متد استاتیک برای محاسبه میانگین متحرک (moving average) لیست داده‌ها است.

get_GINI(self): متدی برای محاسبه میزان خلوص GINI یک نود است.

best_split(self): متدی برای یافتن بهترین تقسیم برای ساخت درخت تصمیم است. این متد با استفاده از الگوریتم GINI ، بهترین ویژگی و مقدار تقسیم را برای تقسیم بعدی درخت تصمیم مشخص می‌کند.

grow_tree(self): متدی با استفاده از تقسیم‌های بهترین یافت شده، درخت تصمیم را به صورت بازگشتی ایجاد می‌کند.

print_info(self, width=4): این متد برای چاپ اطلاعات مربوط به یک نود درخت تصمیم استفاده می‌شود. اطلاعاتی مانند قاعده تقسیم، خلوص GINI نود، توزیع کلاس‌ها در نود و کلاس پیش‌بینی شده را نمایش می‌دهد.

print_tree(self): این متد درخت تصمیم را از نود فعلی به پایین چاپ می‌کند. ابتدا اطلاعات مربوط به نود جاری را با استفاده از متد print_info چاپ می‌کند و سپس به صورت بازگشتی برای نودهای چپ و راست فراخوانی می‌شود تا درخت به صورت کامل چاپ شود.

predict(self, X:pd.DataFrame): این متد برای پیش‌بینی برچسب‌ها بر اساس ویژگی‌های داده‌های ورودی استفاده می‌شود. ورودی X یک DataFrame است که حاوی مجموعه داده‌ها است. متد به صورت دسته‌بندی برای هر داده ویژگی‌های مربوطه را استخراج کرده و با استفاده از متد predict_obs برای هر داده پیش‌بینی را انجام می‌دهد. نتیجه پیش‌بینی‌ها به صورت یک لیست برگشت داده می‌شود.

predict_obs(self, values: dict) -> int: این متد برای پیش‌بینی برچسب یک نمونه بر اساس مقادیر ویژگی‌های آن استفاده می‌شود. مقادیر ویژگی‌ها در یک دیکشنری با نام

values ارسال می‌شود. متد با استفاده از قاعده تقسیم بهترین ویژگی و مقدار تقسیم، نود مناسب را پیدا کرده و به صورت بازگشتی به سمت برگ‌های درخت حرکت می‌کند تا پیش‌بینی نهایی را براساس کلاس پایانی درخت انجام دهد.

cross_val_score(clf, X, Y, cv=5): این متد برای انجام اعتبارسنجی متقابل با استفاده از تابع cross_val_score از کتابخانه scikit-learn استفاده می‌شود .نماینده مدل است، clf X و Y به ترتیب ویژگی‌ها و برچسب‌ها را نشان می‌دهند و cv=5 تعداد بخش‌بندی‌های مختلف برای اعتبارسنجی متقابل را تعیین می‌کند. این متد برای هر بخش اعتبارسنجی، مدل را با داده‌های آموزشی آموزش می‌دهد و دقت را بر روی داده‌های اعتبارسنجی محاسبه می‌کند. سپس دقت‌های به دست آمده را برمی‌گرداند.

print("Cross-validation scores:", cv_scores): این دستور دقت‌های به دست آمده از اعتبارسنجی متقابل را چاپ می‌کند.

print("Mean cross-validation score:", cv_scores.mean()): این دستور میانگین دقت‌های اعتبارسنجی متقابل را چاپ می‌کند. میانگین دقت‌ها نشان دهنده عملکرد مدل در کل مجموعه داده‌ها است.

Y_pred = clf.predict(X_test): با استفاده از متد predict مدل، برچسب‌های پیش‌بینی شده برای داده‌های تست (X_test) محاسبه می‌شود و در Y_pred ذخیره می‌شود.

accuracy = accuracy_score(Y_test, Y_pred): این دستور دقت پیش‌بینی مدل را بر روی داده‌های تست محاسبه می‌کند. Y_test برچسب‌های واقعی داده‌های تست است و Y_pred برچسب‌های پیش‌بینی شده توسط مدل است.

مقایسه دو الگوریتم :

با توجه به ارزیابی‌هایی که در انتهای هر خروجی آمده است، هر دو الگوریتم به طور مشابه عملکرد خوبی در پیش‌بینی تقلب دارند. دقت (Accuracy) هر دو درخت برابر 0.98 است و میانگین امتیازهای اعتبارسنجی متقابل (Cross-validation scores) نیز برای هر دو 0.9775 است .

تفاوت دو الگوریتم:

معیار آنتروپی و Gini index هر دو برای اندازه‌گیری ترکیب ناخالصی یا عدم قطعیت در مجموعه داده‌ها استفاده می‌شوند. این معیارها به دقت و اهمیت ویژگی‌ها در ساخت درخت تصمیم کمک می‌کنند.

تفاوت اساسی بین آنتروپی و Gini index در روش محاسبه ترکیب ناخالصی است. آنتروپی براساس مفهوم انتروپی در نظریه اطلاعات تعریف می‌شود. آنتروپی بیانگر میزان عدم قطعیت در مجموعه داده است. مقدار آنتروپی بین 0 و 1 است و هرچه این مقدار نزدیک به 1 باشد، ناخالصی داده‌ها بیشتر است.

Gini indexنیز میزان ترکیب ناخالصی را اندازه‌گیری می‌کند، اما از مفهوم ضریب Gini استفاده می‌کند که در رابطه با توزیع فراوانی برچسب‌ها درون مجموعه داده است Gini . indexبین 0 و 1 قرار می‌گیرد و هرچه این مقدار نزدیک به 0 باشد، ناخالصی داده‌ها کمتر است.

در مورد دقت و بیش برازش، درخت تصمیم ممکن است در مجموعه داده‌های آموزشی بیش برازش کند، به این معنی که به نمونه‌های آموزشی خود بسیار خوب برازش کرده و برای داده‌های جدید عملکرد نامطلوبی داشته باشد. برای جلوگیری از بیش برازش، می‌توان از روش‌هایی مانند کم کردن عمق درخت، استفاده از روش‌های تقویت کننده مانند ادغام درخت‌ها (ensemble methods) مانند رندوم فرست (Random Forest) و گرادیان بوستینگ (Gradient Boosting) استفاده کرد.

برای افزایش دقت درخت تصمیم، می‌توان از روش‌های زیر استفاده کرد:
1 .انتخاب ویژگی‌های مناسب: با تحلیل ویژگی‌های مختلف و انتخاب ویژگی‌های مهم و مرتبط با مسئله مورد نظر، دقت مدل می‌تواند بهبود یابد.
2 .تنظیم پارامترها: تنظیم پارامترهای مربوط به درخت تصمیم مانند عمق درخت، تعداد حداکثر برگ‌ها و سایر پارامترها می‌تواند بهبودی در دقت مدل ایجاد کند.
3 .استفاده از روش‌ها برای افزایش دقت مانند رندوم فرست(Random Forest) ، گرادیان بوستینگ (Gradient Boosting) و.XGBoost
4 .جمع‌آوری و استفاده از بیشترین حجم داده‌ها: با داشتن مجموعه داده بزرگتر، مدل می‌تواند الگوهای بیشتری را تشخیص دهد و دقت بیشتری داشته باشد.
5 .استفاده از روش‌های انتخاب ویژگی: با استفاده از روش‌های مانند انتخاب ویژگی‌های مهم (feature selection) یا استخراج ویژگی‌های مهم(feature extraction) ، می‌توان ویژگی‌های مهمتر را شناسایی و استفاده کرد.