به نام خداوند بخشنده مهربان

عنوان پروژه

مسئله دسته بندی با پکیج spark

عنوان درس

داده کاوی

استاد

دکتر محمد کیانی ابری

گردآورنده

سید حسین حسینی دولت‌آبادی

تابستان ۱۴۰۴

دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

چکیده

بیماری‌های قلبی-عروقی از دلایل اصلی مرگ‌ومیر در سراسر جهان محسوب می‌شوند. تشخیص زودهنگام و تعیین دقیق شدت این بیماری‌ها می‌تواند نقش حیاتی در بهبود نتایج درمانی داشته باشد. این پروژه به طراحی و پیاده‌سازی یک سیستم هوشمند برای طبقه‌بندی شدت بیماری قلبی بر اساس داده‌های بالینی بیماران می‌پردازد. چالش اصلی این پژوهش، مدیریت یک دیتاست چندکلاسه با عدم توازن شدید بین کلاس‌هاست که می‌تواند منجر به سوگیری مدل و خطاهای تشخیصی خطرناک شود.

برای غلبه بر این چالش، از فریم‌ورک پردازش توزیع‌شده Apache Spark و کتابخانه یادگیری ماشین آن، SparkML، استفاده شده است. یک Pipeline کامل و سرتاسری برای پیش‌پردازش داده‌ها، مهندسی ویژگی و مدل‌سازی طراحی گردید. تکنیک‌های کلیدی مانند مدیریت داده‌های گمشده با Imputer و مقابله با عدم توازن با وزن‌دهی به کلاس‌ها (Class Weighting) به کار گرفته شد. مدل RandomForestClassifier به عنوان الگوریتم پایه انتخاب و عملکرد آن به صورت کمی و کیفی ارزیابی گردید.

نتایج نشان داد که مدل پایه در تشخیص افراد سالم عملکردی ممتاز دارد اما در تفکیک درجات مختلف بیماری با چالش مواجه است. این گزارش به تحلیل دقیق این نتایج، بررسی نقاط ضعف و قوت مدل و ارائه یک نقشه راه برای بهبودهای آتی از طریق تکنیک‌های پیشرفته‌تر می‌پردازد.

کلیدواژگان :یادگیری ماشین، Apache Spark، SparkML، طبقه‌بندی چندکلاسه، داده‌های نامتوازن، بیماری قلبی، جنگل تصادفی.

فهرست مطالب

۱ .مشخصات سیستم و معماری

۱.۱ .مشخصات سیستم اجرایی(Execution System Specifications)

انجام این پروژه بر روی یک زیرساخت پردازش ابری صورت گرفته است که امکان اجرای سریع و کارآمد عملیات پردازش داده بزرگ را فراهم می‌آورد. مشخصات فنی محیط اجرایی که از متادیتای نوت‌بوک استخراج شده، به شرح زیر است:

- پلتفرم اجرایی:Google Colaboratory
- شتاب‌دهنده سخت‌افزاری: واحد پردازش گرافیکی(GPU)
- نوع NVIDIA Tesla T4: GPU
- پردازنده مرکزی (CPU) :Intel(R) Xeon(R) CPU @ 2.20GHz
- حافظه اصلی(RAM) : 12.7 گیگابایت
- هسته نرم‌افزاری(Kernel) :Python 3
- فریم‌ورک اصلی: Apache Spark نسخه 3.5.1
- کتابخانه‌های پشتیبان:
  - Pandas(برای مدیریت داده‌های کوچک و تبدیل نتایج)
  - Matplotlib & Seaborn(برای مصورسازی و رسم نمودار)

استفاده از فریم‌ورک Apache Spark به دلیل قابلیت‌های پردازش توزیع‌شده و مقیاس‌پذیری افقی آن، انتخابی استراتژیک برای این پروژه بوده است. این انتخاب، پروژه را برای کار با دیتاست‌های بسیار بزرگتر در آینده آماده می‌سازد.

۱.۲ .خلاصه‌ای از معماری و فرآیند آموزش

معماری راهکار ارائه شده بر پایه یک Pipeline یادگیری ماشین ماژولار و سرتاسری در SparkML استوار است. این معماری تضمین می‌کند که تمام مراحل از پیش‌پردازش تا آموزش مدل به صورت یکپارچه و تکرارپذیر اجرا شوند.

نمای کلی معماری فرآیند:

1. ورودی داده(Data Ingestion) : دیتاست خام بیماری قلبی از فایل CSV بارگذاری می‌شود.
2. ماژول پیش‌پردازش(Preprocessing Module) : این ماژول مسئول آماده‌سازی داده‌ها برای مدل‌سازی است و شامل مراحل زیر می‌باشد:
   - پاک‌سازی و تبدیل نوع(Data Cleansing & Casting) : حذف ستون‌های غیرمرتبط و اصلاح نوع داده‌های Boolean.
   - مدیریت داده‌های گمشده(Imputation) : جایگزینی مقادیر Null با میانگین ستون به جای حذف ردیف.
   - کدگذاری و برداری‌سازی(Encoding & Vectorization) : تبدیل ویژگی‌های دسته‌ای به عددی (StringIndexer) و تجمیع آن‌ها در یک بردار واحد(VectorAssembler) .
   - مقیاس‌بندی ویژگی(Feature Scaling) : نرمال‌سازی بردار ویژگی‌ها با StandardScaler.
3. ماژول مدل‌سازی(Modeling Module) :
   - الگوریتم: RandomForestClassifier به دلیل قدرت و انعطاف‌پذیری آن انتخاب شده است.
   - مدیریت عدم توازن :تکنیک وزن‌دهی به کلاس‌ها (Class Weighting) مستقیماً در پیکربندی مدل اعمال شده است.

4. ماژول آموزش و ارزیابی(Training & Evaluation Module) :

- آموزش: Pipeline کامل بر روی داده‌های آموزشی fit می‌شود.
- ارزیابی :عملکرد مدل بر روی داده‌های تست با استفاده از معیارهای F1-Score و Accuracy و همچنین ماتریس درهم‌ریختگی سنجیده می‌شود.

۲ .تحلیل داده و تعریف مسئله

۲.۱ .معرفی دیتاست

دیتاست مورد استفاده در این پروژه، مجموعه داده "Heart Disease Uci" از مخزن یادگیری ماشین UCI است. این دیتاست شامل اطلاعات بالینی از بیماران مختلف است که برای تشخیص بیماری‌های قلبی جمع‌آوری شده‌اند. از میان ۷۶ ویژگی موجود، زیرمجموعه‌ای شامل ۱۴ ویژگی کلیدی که در تحقیقات پیشین به عنوان مهم‌ترین عوامل پیش‌بینی‌کننده شناسایی شده‌اند، مورد استفاده قرار گرفت. این ویژگی‌ها عبارتند از:

- ویژگی‌های دموگرافیک: age, sex
- علائم بالینی: cp (نوع درد قفسه سینه)، trestbps (فشار خون استراحت)، chol (کلسترول)
- نتایج آزمایشگاهی: fbs (قند خون ناشتا)، restecg (نتایج الکتروکاردیوگرام)
- پارامترهای عملکردی: thalch (حداکثر ضربان قلب)، exang (آنژین ناشی از ورزش)

ستون هدف(Target Variable) :

ستون num به عنوان متغیر هدف تعریف شده است. این ستون شدت بیماری را در یک مقیاس ترتیبی از ۰ تا ۴ مشخص می‌کند:

- کلاس 0: فرد سالم (عدم وجود بیماری قابل توجه)
- کلاس‌های 1 تا 4 ::شان‌دهنده درجات مختلفی از تنگی عروق (از خفیف تا شدید)

۲.۲ .چالش عدم توازن داده‌ها(Class Imbalance)

تحلیل توزیع فراوانی کلاس‌ها در ستون هدف، چالش اصلی این پروژه را نمایان می‌سازد. همانطور که در نمودار زیر مشاهده می‌شود، دیتاست به شدت نامتوازن است.

(در این قسمت می‌توان یک نمودار میله‌ای (Bar Chart) از توزیع کلاس‌ها قرار داد)

توزیع فراوانی کلاس‌ها:

- کلاس 0 :۴۹۹ نمونه (~۵۴.۲٪)
- کلاس 1 :۲۳۹ نمونه (~۲۶.۰٪)
- کلاس 2 :۱۰۹ نمونه (~۱۱.۸٪)
- کلاس 3 :۶۷ نمونه (~۷.۳٪)
- کلاس 4 :۱۳ نمونه (~۱.۴٪)

این عدم توازن شدید باعث می‌شود که مدل‌های یادگیری ماشین به طور طبیعی به سمت پیش‌بینی کلاس‌های اکثریت (کلاس ۰ و ۱) سوگیری پیدا کنند و در تشخیص کلاس‌های نادر (به خصوص کلاس‌های ۳ و ۴) که اغلب نشان‌دهنده وضعیت‌های بحرانی‌تر هستند، با شکست مواجه شوند. بنابراین، بخش قابل توجهی از معماری پروژه به مدیریت این چالش اختصاص یافته است.

۳ .جزئیات پیاده‌سازی

این بخش به تشریح جزئیات فنی مراحل پیاده‌سازی از آماده‌سازی داده تا ساخت Pipeline نهایی می‌پردازد.

۳.۱ .آماده‌سازی محیط و داده‌ها

فرآیند آماده‌سازی شامل گام‌های زیر بود:

1. بارگذاری داده :دیتاست با استفاده از spark.read.csv بارگذاری شد. پارامترهای header=True و inferSchema=True برای شناسایی صحیح نام ستون‌ها و نوع داده‌ها تنظیم گردید.

2. پاک‌سازی اولیه :ستون‌های id و dataset که حاوی اطلاعات فراداده‌ای و غیرمرتبط با مدل‌سازی بودند، با دستور ()drop. حذف شدند.

3. اصلاح نوع داده(Type Casting) : ستون‌های fbs و exang که توسط Spark به صورت Boolean شناسایی شده بودند، به Integer تبدیل شدند. این گام برای سازگاری با ترانسفورماتور Imputer که تنها بر روی ستون‌های عددی عمل می‌کند، ضروری بود.

4. محاسبه وزن کلاس‌ها :برای مقابله با عدم توازن، یک ستون classWeight به دیتافریم اضافه شد. مقدار وزن برای هر کلاس i از فرمول زیر محاسبه گردید:

$$W_i = \frac{N}{k \times N_i}$$

۳.۲ .معماری Pipeline : جزئیات فنی

استفاده از Pipeline در SparkML یک رویکرد استاندارد برای ساخت جریان‌های کاری یادگیری ماشین است که تکرارپذیری و مدیریت آسان را تضمین می‌کند Pipeline .نهایی این پروژه از مراحل زیر تشکیل شده است:

1. Imputer:
   - inputCols: لیست تمام ستون‌های عددی.
   - outputCols: نام ستون‌های جدید خروجی (با پسوند imputed_).
   - strategy: "mean"(استراتژی جایگزینی با میانگین).
2. StringIndexer(ویژگی‌ها):
   - برای هر ستون دسته‌ای یک StringIndexer مجزا تعریف شد.

- handleInvalid:"keep" ، این پارامتر تضمین می‌کند که مقادیر Null یا مقادیر جدیدی که در داده تست ظاهر می‌شوند، به عنوان یک دسته جداگانه در نظر گرفته شوند و باعث بروز خطا نشوند.

3. StringIndexer(برچسب):
   - ستون num را به ستون label با ایندکس‌های عددی از ۰.۰ تا ۴.۰ تبدیل کرد.
4. VectorAssembler:
   - inputCols:لیستی از تمام ستون‌های پردازش‌شده (imputed_*و indexed_*).
   - outputCol:"features" ، نام بردار خروجی.
5. StandardScaler:
   - inputCol:."features"
   - outputCol:."scaledFeatures"
6. RandomForestClassifier:
   - featuresCol: "scaledFeatures" (استفاده از ویژگی‌های مقیاس‌بندی شده).
   - labelCol:."label"
   - weightCol: "classWeight"(مهم‌ترین پارامتر برای مدیریت عدم توازن).
   - numTrees:۱۰۰، maxDepth: ۱۰ (به عنوان پارامترهای پایه).
   - Seed: ۴۲ (برای اطمینان از نتایج یکسان در اجراهای مختلف).

۳.۳ .فرآیند آموزش و ارزیابی

فرآیند نهایی آموزش و ارزیابی به شرح زیر انجام شد:

- تقسیم داده(Data Splitting) : دیتافریم پردازش‌شده با استفاده از متد randomSplit(). به دو بخش آموزشی (۸۰٪) و تست (۲۰٪) تقسیم شد. استفاده از seed=42 تکرارپذیری این تقسیم‌بندی را تضمین می‌کند.
- آموزش مدل(Model Fitting) : مدل Pipeline با فراخوانی متد fit(). بر روی داده‌های آموزشی (train_data)آموزش داده شد. در این مرحله، تمام پارامترهای ترانسفورماتورها (مانند میانگین در Imputerیا نگاشت در StringIndexer از روی داده‌های آموزشی یاد گرفته شده و مدل RandomForestنیز آموزش می‌بیند. خروجی این مرحله یک PipelineModelاست که آماده پیش‌بینی است.
- پیش‌بینی(Prediction) : مدل آموزش‌دیده با متد transform(). بر روی داده‌های تست (test_data)اعمال شد تا پیش‌بینی‌ها برای هر نمونه تولید شود.
- ارزیابی(Evaluation) : از کلاس MulticlassClassificationEvaluator برای محاسبه معیارهای عملکرد استفاده شد. با تنظیم پارامتر metricName، معیارهای مختلفی مانند f1(پیش‌فرض)، accuracy، weightedPrecisionو weightedRecallقابل محاسبه هستند.

۴ .نتایج و تحلیل‌ها

۴.۱ .نتایج کمی مدل

معیار ارزیابی	مقدار	توضیحات
F1-Score (Weighted)	0.5140	معیار اصلی .میانگین همساز وزن‌دهی شده Precision و .Recall برای داده‌های نامتوازن معتبرتر است.
Accuracy(دقت)	0.5503	درصد کل پیش‌بینی‌های صحیح. در داده‌های نامتوازن می‌تواند گمراه‌کننده باشد.

۴.۲ .تحلیل کیفی و ماتریس درهم‌ریختگی(Confusion Matrix)

1. تسلط در تشخیص کلاس اکثریت :مدل در شناسایی افراد سالم (کلاس ۰) بسیار موفق عمل کرده است. از مجموع ۶۷ بیمار سالم در مجموعه تست، ۵۸ نفر به درستی تشخیص داده شده‌اند (نرخ تشخیص صحیح ~۸۶٪).
2. چالش در تفکیک کلاس‌های میانی :با کاهش فراوانی کلاس‌ها، توانایی مدل در تشخیص آن‌ها به شدت کاهش می‌یابد.
3. سوگیری به سمت کلاس‌های مجاور :مدل تمایل دارد نمونه‌های کلاس‌های نادرتر را به کلاس‌های شایع‌تر و مجاور نسبت دهد. برای مثال، ۱۲ بیمار از کلاس ۲ به اشتباه به عنوان کلاس ۱ طبقه‌بندی شده‌اند.
4. عدم تشخیص کلاس‌های بسیار نادر :مدل در شناسایی کلاس ۴ (که به دلیل کمبود نمونه در مجموعه تست وجود نداشت) و به طور کلی کلاس‌های با شدت بالا ناتوان است.

۵ .نتیجه‌گیری و پیشنهادات

۵.۱ .جمع‌بندی نهایی

در این پروژه، یک سیستم طبقه‌بندی هوشمند برای تشخیص شدت بیماری قلبی با استفاده از فریم‌ورک Apache Spark با موفقیت طراحی و پیاده‌سازی شد. چالش‌های اصلی پروژه، شامل مدیریت داده‌های گمشده و عدم توازن شدید کلاس‌ها، با استفاده از تکنیک‌های استاندارد و کارآمدی مانند Imputer و Class Weighting به خوبی مدیریت گردید. مدل پایه RandomForestClassifier توانست به یک F1-Score برابر با ۰.۵۱ دست یابد که به عنوان یک نقطه شروع قوی برای تحلیل‌های بیشتر عمل می‌کند.

تحلیل عمیق ماتریس درهم‌ریختگی نشان داد که اگرچه مدل در شناسایی افراد سالم موفق است، اما در تفکیک درجات مختلف بیماری و جلوگیری از خطاهای پزشکی خطرناک (False Negatives) نیازمند بهبودهای قابل توجهی است. این پروژه یک بستر فنی محکم و قابل توسعه برای تحقیقات آینده در این حوزه فراهم کرده است.

۵.۲ .پیشنهادات برای کارهای آینده

برای ارتقاء عملکرد مدل و افزایش قابلیت اطمینان آن برای کاربردهای بالینی، نقشه راه زیر پیشنهاد می‌شود:

1. بهینه‌سازی ابرپارامترها(Hyperparameter Tuning) :
   - استفاده از ماژول CrossValidator در SparkML برای جستجوی شبکه‌ای (ParamGridBuilder)و یافتن بهترین ترکیب از پارامترهای numTrees, maxDepthو impurityبرای مدل .RandomForestاین کار می‌تواند به بهینه‌سازی توازن بین بایاس و واریانس مدل کمک کند.
2. آزمایش الگوریتم‌های پیشرفته‌تر:
   - پیاده‌سازی و مقایسه عملکرد با مدل GBTClassifier (Gradient-Boosted Trees)که اغلب در داده‌های جدولی عملکرد بهتری دارد. با توجه به محدودیت این مدل برای مسائل دو کلاسه، باید از یک Wrapper مانند OneVsRestبرای تعمیم آن به مسئله چند کلاسه استفاده کرد.

موفق باشید