به نام خدا

دانشکده‌ی مهندسی برق و کامپیوتر

درس هوش مصنوعی

تمرین کامپیوتری شماره ۴

استاد: دکتر فدایی

علی پاکدل صمدی

۸۱۰۱۹۸۳۶۸

هدف پروژه:

هدف از انجام این پروژه آشنایی با روش‌های machine learning می‌باشد. در این پروژه با کتابخانه Scikit-learn آشنا می‌شویم و با استفاده از آن و machine learning، به تجزیه و تحلیل داده‌ها می‌پردازیم.

توضیح کلی پروژه:

در این پروژه ما یک فایل داده در اختیار داریم که ابتدا داده را بررسی و تجزیه و تحلیل می‌کنیم، سپس آن‌را پیش‌پردازش کرده و با چند روش machine learning به طبقه‌بندی و پیش‌بینی می‌پردازیم، سپس با random forest این‌کار را انجام داده و نتایج را تحلیل می‌کنیم.

فاز صفر: EDA and Visualization

۳. توزیع مقادیر عددی از توزیع گاوسی پیروی می‌کند.

فاز اول: Preprocessing

۱. روش‌های موجود برای مدیریت داده‌های گمشده:

- **پاک‌کردن ردیف:** در این روش اگر خانه‌ای در سطری وجود داشته باشد که مقدارش گمشده باشد، آن سطر را پاک می‌کنیم. این روش هنگامی می‌تواند مفید باشد که داده‌های زیادی در اختیار داشته باشیم و پاک کردن برخی از آنها مشکل‌ساز نباشد. در این روش با حذف برخی سطرها، ممکن است که به پیش‌بینی مورد انتظار خود نرسیم و نتایج متفاوت خواهند شد.

مزایا:

- روش آسانی است و نیاز به انجام عملیات خاصی ندارد.
- این روش می‌تواند یک داده قوی بسازد زیرا معمولا سطر و یا ستون‌های حذف شده اطلاعات مفیدی به ما نمی‌دادند.

معایب:

- اطلاعات و داده ما از دست می‌روند و ادامه روند بدون آنها انجام خواهد شد.

- اگر داده‌های زیادی حذف شوند، دیتا ما ضعیف خواهد شد و به همین دلیل به نتایج ضعیفی نیز خواهیم رسید.

- **جایگزین کردن مقادیر با آماره‌ها:** در این روش می‌توانیم خانه‌هایی که ویژگی عددی دارند را با میانگین یا میانه یا یک ویژگی دیگر جایگزین کرده و ادامه روند را با این مقادیر جلو ببریم. این مقادیر می‌توانند به طور تقریبی یک داده نسبتا خوب دهند و این روش از روش قبلی بهتر عمل می کند. در ویژگی‌های غیر عددی نیز می‌توان از مد برای جایگزینی استفاده کرد.

مزایا:

- در این روش ما داده‌ها را پاک نمی‌کنیم و سعی می‌کنیم آنها را به عدد واقعیشان نزدیک کنیم، به همین دلیل نتایج بهتری نسبت به روش قبل می‌گیریم.
- این روش پیاده‌سازی سختی ندارد و در داده‌هایی که حجم کمی دارند بسیار منطقی می‌باشد.

معایب:

- مقادیر دقیق نخواهند بود و به همین دلیل نتایج ما نیز دقیق نیستند.
- این روش می‌تواند باعث ایجاد سوگیری در نتایج شود.

۲. در این پروژه، ما با استفاده از میانگین برای ویژگی‌های عددی و مد برای ویژگی‌های غیرعددی، داده‌های از دست رفته را جایگزین می‌کنیم.

۳.

- **Normalization:**

نرمال‌سازی اعداد را به بازه ۰ و ۱ اسکیل می‌کند. نرمال‌سازی زمانی مفید است که ویژگی‌ها برروی بزرگی مقادیر آنها متکی است و همچنین داده‌ها مقیاس‌های متفاوتی داشته باشند و شما توزیع داده‌های خود را نمی‌دانید. به عنوان مثال در فاصله در KNN و همچنین شبکه‌های عصبی. در این روش هر داده با رابطه $x = \frac{x_{old}-min}{max-min}$ بدست می‌آید.

- **Standardization:**

استانداردسازی به تغییر توزیع هر ویژگی به سمت داشتن میانگین صفر و انحراف معیار یک اشاره دارد. این روش هنگامی مفید است که ویژگی‌ها برروی یک توزیع مانند توزیع گاوسی متکی باشند. استانداردسازی فرض می‌کند که داده‌های شما دارای توزیع گاوسی است و اگر توزیع ما گاوسی باشد، این روش موثرتر است. در این روش هر داده با رابطه $x = \frac{x_{old}-mean}{std}$ بدست می‌آید.

۴. در این پروژه ما از استانداردسازی استفاده می‌کنیم زیرا داده‌های ما از توزیع نرمال پیروی می‌کنند. اگر در داده‌ها ما نوع توزیع را نمی‌دانستیم، بهتر بود از نرمال‌سازی استفاده می‌کردیم.

۵. روش‌های موجود برای encode کردن داده‌های دسته‌ای:

- Ordinal Encoding:

این روش هنگامی استفاده می‌شود که داده دسته‌ای مورد نظر ترتیبی باشد و یک نوع تربیت در خود داشته باشند و برای ویژگی‌هایی که ترتیب مهم نباشد، از روش‌های دیگر باید استفاده شود. این روش هر داده را با یک عدد جایگزین کرده و هر جا این داده تکرار شود، این عدد به جای آن گذاشته می‌شود.

- One-hot Encoding:

هنگامی که داده دسته‌ای مورد نظر هیچ ترتیبی ندارد(مانند رنگ‌ها)، این روش مفیدتر می‌باشد. در این روش به ازای هر داده در این ویژگی دسته‌ای، یک ستون جدید اضافه می‌شود و به صورت ۰ و ۱ پر می‌شود و در هر سطر که این ویژگی داده مورد نظر را داشت، ۱ قرار می‌گیرد و باقی سطر ها صفر می‌شود. هنگامی که ویژگی دسته‌ای مورد نظر ما تعداد زیادی داده متفاوت داشته باشد، این روش بهینه نخواهد بود زیرا تعداد زیادی ستون به دیتافریم ما اضافه خواهد شد.

- Binary Encoding:

در این روش ما دسته‌ای که دارای تنها دو داده متفاوت می‌باشد(Yes/No یا True/False) از این روش استفاده می‌کنیم و به جای یکی از آنها صفر و به جای دیگری یک قرار می‌دهیم و اینگونه این داده دسته‌ای encode می‌شود.

در این پروژه ما برای key و mode از one-hot encoding استفاده می‌کنیم، زیرا ترتیبی ندارد و اینکه یک vector ایجاد می‌شود که تحلیل آن آسان می‌باشد. همچنین در این دو ویژگی، مقادیر مختلف زیادی وجود ندارد که ستون‌های زیادی اضافه شود. برای artist_name ما از ordinal encoding استفاده می‌کنیم، زیرا مقادیر مختلف زیادی دارد(نزدیک به ۵۰۰۰) و منطقی نمی‌باشد از one-hot encoding استفاده شود.

۶. می‌دانیم که نام خواننده می‌تواند در ژانر آهنگ تاثیر گذار باشد، اما ما تنها داده train را نداریم و می‌خواهیم برروی دیگر داده‌ها نیز پیش‌بینی را انجام دهیم و در آنجا به درد ما نمی‌خورند و باعث overfitting می‌شود. به همین دلیل ما این ستون را نیز بهتر است حذف کنیم.

۸.

- خیر، همه ستون‌ها مفید نیستند و اطلاعات مفیدی به ما نمی‌دهند و تنها باعث می‌شوند زمان و حافظه بیشتری مصرف شود و همچنین overfitting رخ دهد.

- نگه‌داشتن همه ستون‌ها می‌تواند مفید باشد هنگامی که یک ستون information gain کمی در داده train ما داشته باشد اما در واقعیت می‌دانیم این داده مفید خواهد بود و همچنین می‌تواند از underfitting جلوگیری کند. از بدی‌های این روش نیز می‌توان به overfitting و زمان و حافظه مصرفی بالا، پیچیدگی بیشتر اشاره کرد.
- بله می‌توانیم تعدادی از ستون‌ها را حذف کنیم.
- از مزایای حذف ستون‌ها می‌توان به جلوگیری از overfitting و همچنین زمان و حافظه مصرفی کمتر اشاره کرد، باعث افزایش عملکرد و دقت نیز می‌تواند بشود، اما از معایب آن می‌توان به آن اشاره کرد که ممکن است ستونی حذف شود که در train ما به‌درد بخور نبوده و در واقعیت ویژگی مهمی باشد، همچنین می‌تواند باعث ایجاد underfitting شود.
- با توجه به information gain بدست‌آمده می‌توان فهمید که ویژگی‌هایی از قبیل popularity, acousticness, danceability, duration_ms, energy, … می‌توانند مفید باشند.
- با توجه به نتایج می‌توانیم ستون‌های key, mode, tempo, liveness را حذف کنیم، زیرا information gain پایینی دارند.

ما ستون track_name را حذف می‌کنیم، زیرا نام آهنگ‌ها اطلاعات مفیدی درباره ژانر آهنگ نمی‌توانند به ما بدهند و به همین دلیل بهتر است این ستون حذف شود.

فاز دوم: Model Training, Evaluation and Hyper Parameter Tuning

۱. تعدادی از داده‌ها باید به عنوان train و تعدادی به عنوان test در نظر گرفته شوند. روش‌های مختلفی برای تعیین مقدار درصد هرکدام وجود دارد. معمولا 75-25 ,80-20 , 70-30 در نظر گرفته می‌شود.

اگر داده‌های زیادی را به train اختصاص دهیم، باعث overfitting می‌شود و بالعکس اگر داده‌های کمی را به train اختصاص دهیم، underfitting رخ خواهد داد. ما در این پروژه از 70-30 استفاده می‌کنیم.

داده‌های انتخابی باید تصادفی باشند، زیرا ممکن است داده‌ها براساس ترتیب خاصی چیده شده باشند و این باعث شود train انتخابی ما جهت‌گیری نسبت به برخی ژانرها داشته باشد. معمولا خوب است که نسبت تعداد هر ژانر در train و test تقریبا برابر باشد.

استفاده از تقسیم برابر train و test هنگامی می‌تواند مفید باشد که ما دیتا ما بسیار زیاد باشد و از آنجایی که این حالت به ندرت رخ می‌دهد، این روش مناسب نمی‌باشد.

Stratify باعث می‌شود train و test از برچسب هدف، به نسبتی داشته باشند که ارائه شده است.

۲. ما به صورت 70-30 و با random ۴۲ تقسیم کردیم.

۳. برای این مدل همانطور که مشاهده می‌شود، train دقت بالایی دارد ولی test دقت کم. این می‌تواند نشان دهد که کمی overfitting رخ داده است اما به دلیل اینکه دقت‌ها در حدود ۶۰ هستند، اورفیتیگ خیلی کمی رخ داده است و بهترین دقت برای test در k=15 می‌باشد.

۴. با هایپرپارامتر max-depth تا عمق ۶ و ۷، train و test تقریبا منطبق هستند و صعودی. سپس train به دقت بالایی می‌رسد و test کمی نزولی می‌شود که نشان از overfitting می‌دهد اما به دلیل اینکه دقت‌ها در حدود ۶۰ هستند، اورفیتیگ خیلی کمی رخ داده است. بهترین max-depth نیز ۱۰ می‌باشد.

با هایپرپارامتر min-samples-leaf، train نزولی می‌باشد و test به صورت صعودی، در واقع رفته رفته با underfitting نزدیک می‌شویم اما به دلیل صعودی بودن دقت test، می‌توان گفت uderfitting بسیار کمی در حال رخ دادن است. بهتری min-samples-leaf نیز ۱۴ می‌باشد.

۵.

Overfitting: این حالت زمانی رخ می‌دهد که مدل ما train را به حد زیادی با دقت بالایی بیاموزد. در واقع آموزش train به حدی زیاد است که باعث تاثیر منفی در کل مدل می‌شود. این حالت موقعی رخ می‌دهد که برخی ویژگی‌هایی که مهم نیستند و خاص آن train هستند(نویز)، برروی مدل ما اثر بگذارد. یکی از جاهایی که این حالت رخ می‌دهد، این است که در split داده‌ها، به داده‌های train مقادیر زیادی را دهیم. این حالت واریاس بالا و سوگیری کمی دارد.

Underfitting: این حالت زمانی رخ می‌دهد که دقت در train و هم در test پایین باشد و هر دو داده را نتوانیم خوب ارزیابی کنیم. در این حالت مدل ما نمی‌تواند رابطه بین متغیرهای ورودی و خروجی را به طور دقیق ثبت کند و عملکرد ضعیفی نشان می‌دهد. اگر ما داده‌های کمی را به عنوان train در split داده‌ها در نظر بگیریم، می‌تواند underfitting رخ دهد. این حالت دارای واریانس کم و سوگیری بالا می‌باشد.

از آنجایی که دقت در بهترین حالت‌ها برای train بالای ۷۰ و برای test در حدود ۶۵ می‌باشد، می‌توان گفت هیچ‌کدام از آنها رخ نداده است اما نمی‌توان گفت که بهترین مدل سازی است و با کمی تغییر می‌تواند به overfitting و یا underfitting نزدیک شود.

۶.

- Recall:

Recall به این معنی است که چه تعداد از true positive ها پیدا شده است. در واقع نسبت پیش‌بینی‌های true positive می‌باشد. Recall معیاری برای کمیت می‌باشد.

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

- Precision:

Precision نسبت true positive به کل مشاهدات positive می‌باشد. دقت معیاری برای کیفیت می‌باشد.

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

- Accuracy:

دقت نسبت پیش‌بینی‌های درست به کل پیش‌بینی‌ها می‌باشد. دقت به تنهایی نمی‌تواند نشان از خوب یا بد بودن مدل بدهد. اگر false positive و false negative ها متقارن نباشند، دقت به تنهایی کافی نخواهد بود.

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

- F1-Score:

امتیاز F1 میانگین هارمونیک precision و recall میباشد. این امتیاز بین precision و recallمیباشد. این امتیاز ترکیبی از precision و recall میباشد و برای تحلیل نتایج مفیدتر میباشد. حداکثر مقدار 1 دارد که هرچه precision و recall بالا باشند این نیز به یک میل میکند و حداقل مقدار 0 دارد که هنگامی که یکی از آن دو صفر باشد، آن مقدار را میگیرد .این امتیاز یک میانگین وزندار از precision و recall میباشد و هم false positive و هم false negativeها در میانگین خود حساب میکند. این امتیاز معمولا از accuracy مفیدتر می‌باشد به خصوص وقتی که یک توزیع نابرابر داشته باشیم.

$$f1 - score = \frac{2(recall \times precision)}{recall + precision}$$

۷. با توجه به همان مزایا و معایبی که در بالا گفته شد، می‌توان فهمید که اگر ما در مدیریت گمشده‌ها، داده‌هارا حذف می‌کردیم، تعداد داده‌ها کاهش می‌یافت و کمبود داده در machine learning مضر می‌باشد. همچنین در این روشی که پیش‌گرفتیم، یک سوگیری به سمت مد و میانگین داشته‌ایم. همچنین در حذف برخی ستون‌ها، اگر ما به عنوان مثال ستون track_name را حذف نمی‌کردیم، باعث ایجاد اورفیتیگ می‌شد، زیرا این یک ویژگی است که فاقد اهمیت در واقعیت است و تنها باعث ایجاد یک ویژگی خاص برای آن سطر می‌شود(نویز). همچنین با حذف ستون‌های کم اهمیت‌تر، باعث افزایش سرعت شدیم.

فاز سوم: Ensemble Methods

۲.

- **Max_depth:** عمق حداکثر یا تعداد splitهایی است که می‌خواهد درخت عمیق شود. یعنی به تعداد این عدد درخت‌ها در هر شاخه split می‌شوند. این باعث می‌شود train دیتا ما کمتر آموزش ببیند و به همین دلیل واریانس کم می‌شود و خطا نیز کاهش پیدا می‌کند.
- **N_estimator:** تعداد درختان می‌باشد. معمولا هر چه تعداد درختان بیشتر شود، میانگین بهتری می‌توان از این درخت‌ها گرفت. اما این باعث می‌شود هزینه محاسباتی بسیار زیاد شود. در بعضی موارد اگر تعداد درختان بیشتر از حدی شود، دقت پایین آمده و عملکردش لزوما صعودی نمی‌باشد.
- **Min_samples_leaf:** این هایپرپارامتر نشان‌دهنده حداقل تعداد نمونه برای قرار گرفتن در یک گره برگ می‌باشد.

نتیجه‌گیری کلی:

در این پروژه با انواع روش‌های machine learning و کتابخانه scikit-learn آشنا شدیم. در این پروژه فهمیدیم تمام ویژگی مفید نیستند و باید از آنهایی که مهم هستند و به ما اطلاعات می‌دهند استفاده شود. همچنین اینکه با روش split دیتا آشنا شدیم و توانسیتم با KNN و Decision Tree و Random Forest و هایپرپارامترهای آنها به پیش‌بینی test خود بپردازیم و دیدیم که Random Forest می‌تواند نتایج بهتری به باقی دهد.

منابع استفاده شده:

لینک‌های داده شده در PDF

https://stackoverflow.com

https://www.geeksforgeeks.org

https://www.wikipedia.org/

https://scikit-learn.org/