گروه 31

پروژه 2:

لینک گوگل کولب

https://colab.research.google.com/drive/1OoNTJPnoXJWwPW2QGQEduNDVLYqsUxOi#scrollTo=1uCZO7P_0695

گزارش:

مرحله اولیه و بسیار اساسی در فرایند تحلیل داده، EDA یا تحلیل اکتشافی داده است. EDA ابزاری قدرتمندی است که به ما این امکان را می‌دهد تا اطلاعات اولیه از داده‌ها به دست آوریم و الگوها، ارتباطات و نقاط قوت و ضعف داده‌ها را شناسایی کنیم و در ادامه بتوانیم از مدل‌های یادگیری ماشین برای دسته‌بندی بهینه استفاده کنیم.

در ابتدا تمامی کتابخانه‌های مورد نیاز را وارد می‌کنیم . در ادامه نیز هر کجا متوجه شدیم نیاز به وارد کردن کتابخانه‌ی جدید است، آن را نیز اضافه می کنیم .

در ادامه، مسیر فایل با فرمت CSV که می‌خواهیم اطلاعات آن را بخوانیم را در متغیر file_path ذخیره می‌کنیم.

سپس با استفاده از تابع read_csv از کتابخانه pandas ، فایل را می خوانیم. در ادامه این داده‌ها را در یک ساختار داده به نام data ذخیره می‌کنیم. این ساختار می‌تواند به عنوان یک جدول دوبعدی از داده‌ها در نظر گرفته شود.

سپس تابع head() سطرهای ابتدایی ساختار داده data را نمایش می‌دهد .

در ادامه 3خط را به طور تصادفی و بدون امکان تکرار چاپ می‌کنیم.

با تابع np.random.choice()، اندیس‌های تصادفی برای انتخاب ردیف‌ها ایجاد می‌شود. پارامتر size تعیین کننده تعداد اندیس‌های تصادفی است و با استفاده ازreplace=False ردیف‌ها غیرقابل تکرار خواهند بود. سپس با تابع ilocبه محتوای ردیف‌های انتخاب شده دسترسی پیدا کرده و آن ها را چاپ می‌کنیم.

سپس اطلاعات کلی دیتاها را مشاهده می‌کنیم که به طور مثال شامل نوع داده و تعداد داده های غیرnullدر هر ستون می‌باشد.

در بخش بعدی ویژگی هایی شامل تعداد، میانگین، انحراف معیار، حداقل، چارک‌ها و حداکثر را بررسی می‌کنیم. به این صورت اطلاعات دقیق‌تری از دیتاست پیدا کرده و میتوانیم حدود داده‌ها و داده‌های پرت را بررسی کنیم.

سپس برای بررسی توزیع در داده هدف از یک نمودار countplot استفاده می‌کنیم ، زیرا توازن یا ناتوازن کلاس‌ها می‌تواند بر توانایی مدل در یادگیری و پیش‌بینی تأثیر بگذارد.

این نموار که تعداد yes و no را در ستون هدف بررسی می کند نشان می دهد تعداد no ها بیشتر است.

نمودار هیستوگرام برای ویژگی‌های مختلف است که اطلاعاتی در مورد توزیع و توازن ویژگی‌ها در دیتاست ارائه می‌دهد. هر هیستوگرام نمایانگر توزیع‌های مختلفی است که در داده وجود دارد.

سپس از نمودار جعبه‌ای (Box Plot) استفاده می‌کنیم تا توزیع دما را در زمان مشخصی (ساعت 9 صبح) بررسی کنیم. این نوع نمودار به ما اطلاعات مفیدی از قبیل میانه، چارک‌ها، و نقاط دورافتاده (outliers) می‌دهد.

در ادامه همین نمودار را برای ساعت 3 بعدازظهر بررسی می‌کنیم. متوجه میشویم که دما در زمان دوم بیشتر است . (می‌توان از طریق میانه مقایسه کرد)

در نمودار میله‌ای (Bar Plot) برای نمایش میانگین دمای حداکثر در ایستگاه‌های مختلف هواشناسی استفاده می‌کنیم. این نوع نمودار به ما کمک می‌کند تا مقایسه‌ای بین میانگین دماهای مختلف در ایستگاه‌های مختلف انجام دهیم.

در نمودار بعد، تغییرات دما در بازه‌های زمانی مختلف را بررسی میکنیم . برای جلوگیری از تراکم نمودار نیز می‌توان بازه های زمانی ماهانه یا سه ماهه را در نظر گرفت.

سپس به سراغ داده‌های گمشده می‌رویم . ابتدا تعداد آن‌ها را در هر ستون بررسی می‌کنیم.

سپس ستون‌های بلا استفاده را حذف می‌کنیم و اقدام به پر کردن مقادیر گمشده می‌نماییم.

داده‌های عددی با میانگین مقدارشان و داده‌های دسته‌ای با بیشترین مقدار (mode) تکمیل می‌شوند. در نهایت، تعداد مقادیر نال در هر ستون چاپ می‌شود تا اطمینان حاصل شود که داده‌ها به درستی تکمیل شده‌اند.

کد بعدی برای تبدیل مقادیر غیر عددی در DataFrame به مقادیر عددی با استفاده از LabelEncoder از کتابخانه scikit-learn است. این داده‌های دسته‌ای (categorical) را برای استفاده در مدل‌های ماشین لرنینگ آماده می‌کند.

بخش‌های مختلف:

تبدیل ستون تاریخ به نوع تاریخ و تنظیم آن به عنوان ایندکس: این کار برای اطمینان از این است که داده‌ها به درستی نمونه‌گیری و تحلیل شوند.

تکمیل داده‌های ناقص: این فرآیند تضمین می‌کند که هیچ داده‌ی ناقصی در DataFrame وجود ندارد.

استفاده از LabelEncoder: این کار داده‌های دسته‌ای را به مقادیر عددی تبدیل می‌کند که برای مدل‌های یادگیری ماشین مناسب هستند.

سپس چند سطر اول از داده‌ها نمایش داده می‌شود تا بتوان تغییر مقادیر را مشاهده کرد.

(در پیوست یک تفاوت دو روش برای تبدیل داده‌های غیر عددی به داده‌های عددی بررسی می‌شود.)

حال به سراغ حذف داده های پرت می‌رویم و از روشIQRاستفاده می‌کنیم. برای شهود بهتر، یک بار نمودار جعبه ای را قبل از حذف داده های پرت و یک بار بعد از حذف آن ها رسم می‌کنیم . به وضوح تغییر در داده های پرت مشاهده میشود.

سپس ماتریس کرولیشن را رسم می‌کنیم.

ماتریس همبستگی نشان دهنده‌ی تاثیر ویژگی‌های مختلف روی یکدیگر است و باعث درک ارتباط بین آن‌ها می‌شود. این اطلاعات می‌توانند در انتخاب ویژگی‌ها یا حتی تفسیری از رفتار مدل مفید باشند.

در این ماتریس مقادیر نزدیک به ۱ یا 1- نشان‌دهنده ارتباط قوی مثبت یا منفی است.

اگرعدد نشان داده شده در هر بخش مربعی، عددی مثبت و نزدیک به 1 باشد ، نشان دهنده یک ارتباط مثبت بزرگ است. یعنی این که دو ویژگی با هم همبستگی بسیار زیادی دارند و اگر که یکی از آنها افزایش پیدا کند، دیگری نیز افزایش پیدا می‌کند و برعکس.

اگر منفی و نزدیک به 1- باشد ،نشان دهنده یک ارتباط منفی بزرگ است. این به معنای این است که این دو ویژگی نیز با هم همبستگی بسیار زیادی دارند ولی این بار هنگامی که یکی از آنها افزایش می‌یابد، دیگری کاهش می‌یابد و برعکس، یعنی مخالف هم هستند.

اگر عدد، مثبت یا منفی و نزدیک به 0 باشد به معنای این است که دو ویژگی با هم همبستگی دارند ولی روی هم تاثیر کمی میگذارند و اگر عدد مثبت باشد تاثیر مستقیم و اگر منفی باشد تاثیرش عکس است.

بدیهی است که در سلول متناظر هر متغیر با خودش عدد 1 را می‌بینیم، یعنی بیشترین تاثیر گذاری .

در ماتریس اول رسم شده ، چند سطر و ستون کاملا سفید می‌بینیم . سه ردیف خالی (سفید) در ماتریس همبستگی به این دلیل است که این ویژگی‌ها در داده‌ها به احتمال زیاد شامل مقادیر ثابت زیاد یا دارای واریانس صفر هستند. این ویژگی‌ها نمی‌توانند همبستگی مناسبی با سایر ویژگی‌ها داشته باشند زیرا تنوع داده‌ها برای محاسبه همبستگی وجود ندارد.

برای حل این مشکل قبل از محاسبه ماتریس همبستگی، میتوان ویژگی‌هایی که واریانس صفر یا بسیار کم دارند را حذف کرد. این کار را می‌توان با استفاده از تابع VarianceThreshold|از کتابخانه scikit-learnانجام داد.

بعد از انجام این کار مجددا ماتریس را رسم می‌کنیم و ویژگی هایی که همبستگی بالایی با یکدیگر دارند (با تعیین یک آستانه همبستگی) را حذف می‌کنیم. باید دقت کنیم که در محاسبات، همبستگی یک ستون با خودش در نظر گرفته نشود.

همبستگی بالا بین ویژگی‌ها می‌تواند منجر به افزونگی (redundancy) شود و مدل‌ها را ناپایدار یا کم دقت کند.

استفاده از setبرای ویژگی‌های حذف شده تضمین می‌کند که هر ویژگی تنها یک بار حذف شود.

آستانه‌ی همبستگی یک هایپرپارامتر است که با توجه به ویژگی های مسئله و دیتاست باید تعیین شود و مقدار 0.9 برای همبستگی ،مقدار بالایی می‌باشد.

سپس به سراغ پیاده سازی مدل ها می رویم.

ابتدا ستون هدف را جدا میکنیم.

سپس برای توازن داده‌های نامتوازن از تکنیک SMOTE (Synthetic Minority Over-sampling Technique) استفاده میکنیم.

پارامتر sampling_strategy='auto' به این معنی است که نمونه‌برداری برای توازن تمام کلاس‌ها انجام خواهد شد.

در نهایت نیز تعداد نمونه‌های هر کلاس پس از اعمال SMOTE نمایش داده میشود. بعد از اجرای این کد، داده‌ها توازن یافته و تعداد نمونه‌های هر کلاس برابر خواهد شد.

همانطور که در ابتدا گفته شد داده‌های نامتوازن مشکلی رایج در بسیاری از مسائل یادگیری ماشین است که می‌تواند عملکرد مدل را تحت تأثیر قرار دهد SMOTE .یکی از تکنیک‌های oversampling است که با تولید نمونه‌های مصنوعی از داده‌های اقلیت به توازن داده‌ها کمک می‌کند و باعث بهبود عملکرد مدل‌های یادگیری ماشین می‌شود.

در ادامه داده ها را استاندارد می‌کنیم.

استانداردسازی یکی از مراحل پیش‌پردازش مهم در مسائل یادگیری ماشین است. این تکنیک مقادیر ویژگی‌ها را به گونه‌ای تبدیل می‌کند که میانگین آن‌ها برابر با 0 و انحراف معیار آن‌ها برابر با 1 شود. این کار به مدل‌های یادگیری ماشین کمک می‌کند تا عملکرد بهتری داشته باشند، به ویژه برای مدل‌هایی که مبتنی بر گرادیان هستند (مانند رگرسیون لجستیک، شبکه‌های عصبی و ...).

متد fit_transform برای اعمال استانداردسازی به داده‌های X_res استفاده می‌شود. این متد دو عملیات را به صورت متوالی انجام می‌دهد:

fit: محاسبه‌ی میانگین و انحراف معیار برای هر ویژگی در داده‌هایX_res.

transform : استانداردسازی داده‌ها با استفاده از میانگین و انحراف معیار محاسبه شده.

بعد از اجرای این کد، داده‌های X_resکه قبلاً با استفاده از تکنیک SMOTE توازن یافته‌اند، استانداردسازی می‌شوند و در متغیر X_res_scaledذخیره می‌گردند.

سپس کاهش بعد داده‌ها با استفاده از روش تحلیل مولفه‌های اصلی (PCA) را انجام می‌دهیم.

تحلیل مولفه‌های اصلی (PCA) یک تکنیک کاهش بعد است که به کاهش پیچیدگی داده‌ها و افزایش کارایی مدل‌های یادگیری ماشین کمک می‌کند. این تکنیک با پیدا کردن مولفه‌های اصلی، ابعاد داده‌ها را کاهش می‌دهد و اطلاعات اصلی را حفظ می‌کند. برخی از مزایای استفاده از PCA عبارتند از:

کاهش ابعاد داده‌ها و در نتیجه کاهش زمان محاسباتی مدل‌ها.

حذف نویز و کاهش همبستگی بین ویژگی‌ها.

افزایش دقت مدل‌های یادگیری ماشین با کاهشoverfitting .

ابتدا یک شیء PCAایجاد می‌کنیم و تعداد مولفه‌هایی که باید حفظ شوند را تعیین می‌کنیم. پارامتر n_components مشخص می‌کند که داده‌ها به چند مولفه اصلی کاهش داده شوند.

متد fit_transform برای اعمال PCA به داده‌های X_res_scaledاستفاده می‌شود. این متد دو عملیات را به صورت متوالی انجام می‌دهد:

محاسبه‌ی مولفه‌های اصلی از داده‌ها X_res_scaled.

تبدیل داده‌ها به فضای مولفه‌های اصلی جدید.

بعد از اجرای این کد، داده‌های استانداردسازی شده‌ی X_res_scaledبه فضای مولفه‌های اصلی کاهش داده می‌شوند و در متغیر X_res_pcaذخیره می‌گردند. این داده‌های کاهش بعد یافته می‌توانند به عنوان ورودی به مدل‌های یادگیری ماشین استفاده شوند تا عملکرد بهتری حاصل شود.

هایپرپارامتر n_components در PCA :

n_componentsدر PCA تعیین می‌کند که چند مؤلفه اصلی (Principal Components) باید حفظ شوند. انتخاب مقدار مناسب برای این هایپرپارامتر بسیار مهم است، زیرا:

مقادیر کوچک :انتخاب تعداد کمی مؤلفه ممکن است به از دست رفتن اطلاعات مهم و کاهش عملکرد مدل منجر شود.

مقادیر بزرگ :انتخاب تعداد زیادی مؤلفه ممکن است منجر به حفظ نویز و افزایش پیچیدگی مدل شود، که می‌تواند به overfitting منجر شود.

روش‌های مختلفی برای تنظیم n_componentsوجود دارد:

انتخاب یک عدد ثابت :مانند 15. این روش ساده است ولی ممکن است بهترین نتیجه را ندهد.

نسبت واریانس :انتخاب n_componentsبه گونه‌ای که یک درصد مشخصی از واریانس کل داده‌ها را توضیح دهد.

جستجوی هایپرپارامتر :استفاده از روش‌هایی مانند جستجوی شبکه‌ای یا جستجوی تصادفی برای پیدا کردن بهترین مقدار.

به عنوان اولین مدل ، داده‌ها را به مجموعه‌های آموزشی، اعتبارسنجی و تست تقسیم می‌کنیم و یک مدل SVM (Support Vector Machine)را پیاده‌سازی می‌کنیم.

ابتدا داده‌ها به دو مجموعه تقسیم می‌شوند: 70٪ برای آموزش و 30٪ برای مجموعه موقت.(X_temp, y_temp)

سپس مجموعه موقت به دو قسمت مساوی تقسیم می‌شود: 15٪ برای اعتبارسنجی و 15٪ برای تست.

یک مدل SVM با کرنل خطی ایجاد و آموزش داده می‌شود.

انتخاب کرنل خطی برای SVM (Support Vector Machine) بستگی به ماهیت داده‌ها و مسئله دارد.

سادگی و سرعت

کرنل خطی نسبت به کرنل‌های پیچیده‌تر مانند RBF یا پلی‌نومیک، ساده‌تر و سریع‌تر است.

داده‌های قابل جداسازی خطی

اگر داده‌ها به طور تقریبی به صورت خطی قابل جداسازی باشند، کرنل خطی می‌تواند عملکرد بسیار خوبی داشته باشد. در این صورت نیازی به استفاده از کرنل‌های غیرخطی نیست.

تفسیرپذیری

مدل‌های خطی تفسیرپذیری بهتری دارند. وزن‌های مدل خطی مستقیماً نشان می‌دهند که کدام ویژگی‌ها در تصمیم‌گیری مهم‌تر هستند.

ابعاد بالا

در مسائل با تعداد ویژگی‌های بسیار زیاد (به عنوان مثال، در مسائل متن‌کاوی و ژنومیک)، کرنل خطی معمولاً عملکرد خوبی دارد. این به دلیل خاصیت کرنل خطی در فضاهای با ابعاد بالا است که باعث می‌شود داده‌ها به طور طبیعی جدا شوند.

جلوگیری ازoverfitting

زمانی که داده‌های آموزشی کم باشند استفاده از کرنل‌های پیچیده‌تر می‌تواند منجر به overfitting شود.

تقسیم‌بندی داده ها به سه بخش برای جلوگیری از overfitting میتواند مفید باشد. با استفاده از مجموعه اعتبارسنجی می‌توان مدل را بهینه‌سازی کرد و با استفاده از مجموعه تست عملکرد نهایی مدل را ارزیابی نمود.

سپس ارزیابی مدل را انجام میدهیم . به این منظور ماتریس درهم ریختگی و معیارهای ارزیابی (accuracy، precision، recall، F1) را با استفاده از کتابخانه های آماده محاسبه میکنیم.

در معیار های ارزیابی به دو عبارت زیر برمیخوریم:

macro avg: میانگین دقت، بازخوانی و F1-Score برای هر کلاس به صورت جداگانه محاسبه شده و سپس میانگین گرفته شده است. این معیار برابر با میانگین اعداد به صورت میانگین وزن‌دار نیست.

weighted avg: مشابه میانگین ماکرو است، اما وزن‌دهی به این میانگین‌ها بر اساس تعداد نمونه‌ها انجام می‌شود. این معیار نشان می‌دهد که مدل چقدر کلاس‌های مختلف در مجموعه داده را درست تشخیص داده است.

KNN

برای به کار بردن KNN باید ابتدا مقدار بهینه k را بدست آوریم. در این پروژه اعداد 1 تا 10 را برای k انتخاب می کنیم و هر کدام از k ها که دقت مدل در آن بیشتر بود آن مقدار را در نظر می گیریم. برای این کار از تابع cross_val_score استفاده می کنیم که داده ها و برچسب ها را می گیرد و آرایه ای از دقت ها را با توجه به انتخاب های مختلف از داده های تست و ترین خروجی می دهد. درپارامتر cv=k_folds تعداد بخش های که داده ها به آنها تقسیم می شوند را نشان می دهیم که ما عدد 5 را انتخاب کردیم. وقتی این پرامتر 5 باشد این تابع داده ها را به 5 قسمت مساوی تقسیم می کند و سپس هر بار یکی از این 5 قسمت را به عنوان داده های تست در نظر می گیرد و بقیه را به عنوان داده های ترین در نظر می گیرد سپس دقت های بدست آمده را در یک آرایه خروجی می دهد.

```
k_folds = 5

# لیستی از مقادیر برای پارامتر K که می‌خواهیم امتحان کنیم
k_values = list(range(1, 10))

# لیستی برای نگه‌داری دقت‌های هر مقدار K
accuracies = []

# انجام اعتبارسنجی متقاطع برای هر مقدار K
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_res_scaled, y_res, cv=k_folds, scoring='accuracy')
    accuracies.append(scores.mean())
```

در هر تکرار از این for میانگین دقت های بدست آمده را به آرایه accuracies اضافه می کنیم.

```
# بر اساس دقت میانگین K یافتن بهترین مقدار
best_k = k_values[np.argmax(accuracies)]
best_accuracy = max(accuracies)

print('بهترین مقدار K: ', best_k)
print('دقت متناظر: {:.2f}%'.format(best_accuracy * 100))
```

سپس بهترین مقدار k همراه با دقت حاصل آن را بدست می آوریم.

```
X_train, X_test, y_train, y_test = train_test_split(X_res_scaled, y_res, test_size=0.2, random_state=42)
KNn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)
```

سپس همانطور که در تصویر مشاهده می کنیم داده های متوازن و استاندارد شده را به داده های تست و ترین با ضریب 0.2 تقسیم می کنیم و مدل را با بهترین k (2) ترین می کنیم.

```
accuracy = accuracy_score(y_test, y_pred)
print('دقت مدل: {:.2f}%'.format(accuracy * 100))

print("Train Classification Report:")
print(classification_report(y_test, y_pred))

conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('predicted')
plt.ylabel('actual')
plt.title('confusion matrix')
plt.show()
```

سپس دقت مدل بدست آمده را چاپ می کنیم و همچنین معیار های precision وrecall و f1-score را با استفاده از تابع classification_report که داده های y_test و y_pred که توسط مدل بدست آمده است را با هم مقایسه می کند و سپس ماتریس سردرگمی را برای داده های تست رسم می کنیم. دقت در این مدل 84 درصد است و معیار هایrecall و precision و f1-score به ترتیب برای کلاس 1 94 و 79 و 86 است که بالا بودن معیار recall قابل توجه است.

درخت تصمیم

```
dt = DecisionTreeClassifier()

# آموزش مدل بر روی داده های آموزشی
dt.fit(X_train, y_train)

# پیش‌بینی بر روی داده های آزمون
y_pred = dt.predict(X_test)

# محاسبه دقت مدل
accuracy = accuracy_score(y_test, y_pred)
print('دقت مدل درخت تصمیم: {:.2f}%'.format(accuracy * 100))
```

یک مدل درخت تصمیم با پرامتر های پیش فرض را در نظر می گیریم.البته درخت تصمیم با پارامتر min_samples_split = 10, 20 , 100 و min_samples_leaf = 10,20,100 را امتحان کردیم که در همه حالت ها از حالت پیش فرض دقت کاهش می یابد پس همان حالت پیش فرض را در نظر گرفتیم. در ادامه نیز معیار های recall و precision و f1-score را برای هر دو کلاس چاپ می کنیم را برای کلاس 1 به ترتیب 88 و 86 و 87 خروجی داده می شود. می بینیم که با توجه به KNN این مدل تقریبا f1-score مشابه ای دارد اما این در این مدل هر دو معیار precision و recall متعادل تر هستند و نزدیک به هم هستند. البته هر دو مدل با توجه به svm نتایج بهتری را کسب می کنند(شاید به دلیل اینکه در svm داده های اعتبارسنجی هم داریم)

پیوست 1-

روش‌هایی مانند Label Encoding و One-Hot Encoding برای تبدیل مقادیر دسته‌بندی (categorical) به مقادیر عددی استفاده می‌شوند. با این حال، این دو روش در نحوه تبدیل داده‌ها و کاربردهایشان تفاوت دارند.

Label Encoding

مقادیر دسته‌بندی را به مقادیر عددی تبدیل می‌کند. در این روش، هر دسته به یک عدد منحصر به فرد نگاشت می‌شود. به عنوان مثال، دسته‌ها به این صورت ممکن است نگاشت شوند:

Red -> 0

Green -> 1

Blue -> 2

این روش برای مدل‌هایی که می‌توانند با مقادیر ترتیبی کار کنند، مناسب است. اما در برخی موارد ممکن است باعث بروز مشکل شود، زیرا مدل ممکن است رابطه ترتیبی بین دسته‌ها را فرض کند (مثلاً فرض کند که 2 بزرگ‌تر از 1 است).

One-Hot Encoding

هر دسته را به یک بردار باینری تبدیل می‌کند. در این روش، هر دسته به یک بردار که فقط یک عنصر آن برابر با 1 و بقیه عناصر برابر با 0 هستند، نگاشت می‌شود. به عنوان مثال:

Red -> [1, 0, 0]

Green -> [0, 1, 0]

Blue -> [0, 0, 1]

این روش برای مدل‌هایی مناسب است که با مقادیر ترتیبی کار نمی‌کنند و نمی‌خواهید مدل رابطه ترتیبی بین دسته‌ها فرض کند.

نتیجه‌گیری:

Label Encoding برای مدل‌هایی که با مقادیر ترتیبی به درستی کار می‌کنند مانند درخت‌های تصمیم، جنگل‌های تصادفی و GBM مناسب است.

One-Hot Encoding برای مدل‌هایی که به مقادیر ترتیبی حساس هستند مانند رگرسیون لجستیک، SVM، KNN، Naive Bayes و شبکه‌های عصبی مناسب است.