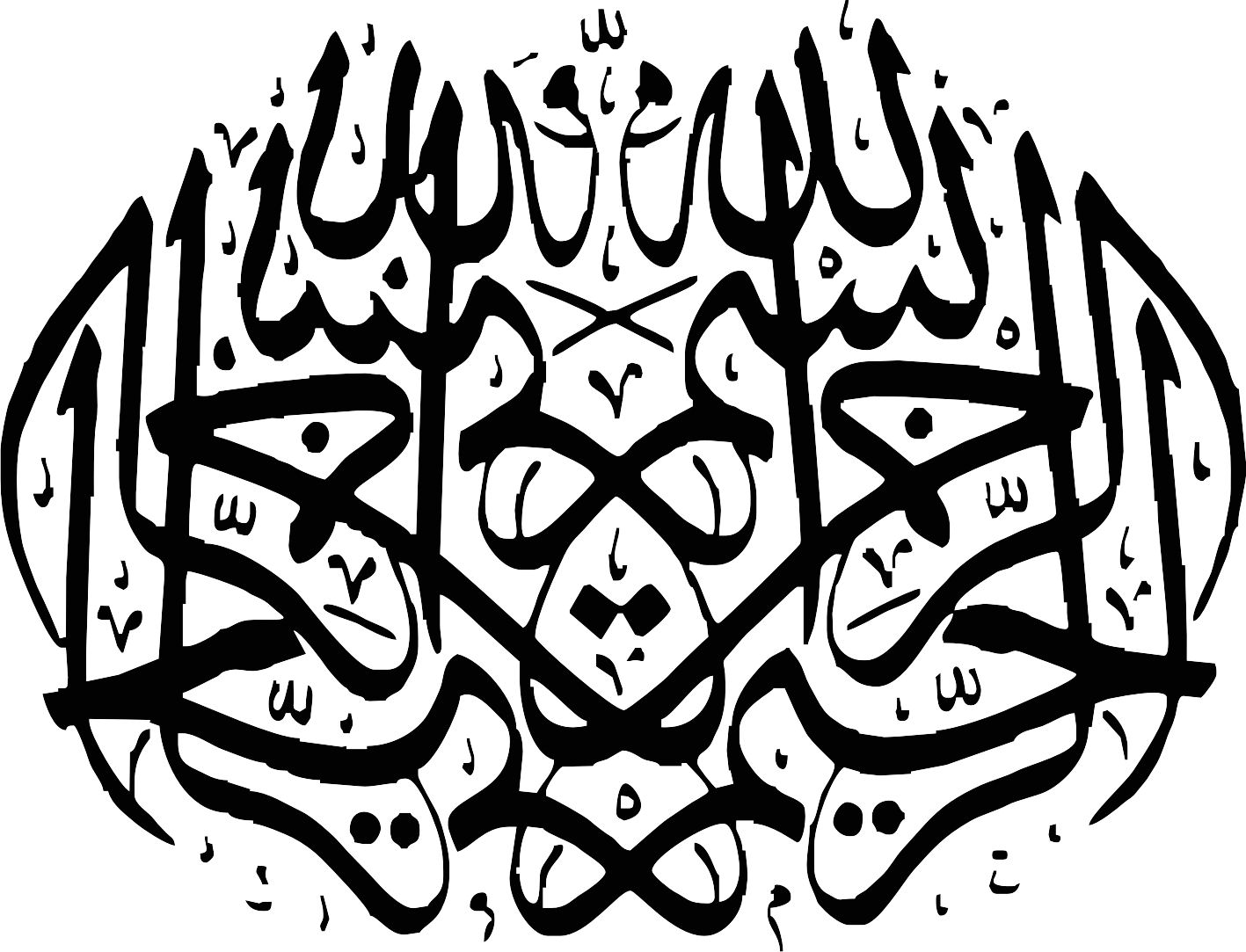


**گزارش تکلیف چهارم درس یادگیری ماشین کاربردی**

**استاد درس: دکتر ناظرفرد**

تهیه کننده: سید نیما محمودیان

شماره دانشجویی: 402125005



فهرست مطالب

[1-پاسخ سوال اول 1](#_Toc165282316)

[1-1-لود کردن دیتاست، نمایش ده ردیف به صورت تصادفی، و بررسی کلی داده‌ها 1](#_Toc165282317)

[2-1-مصور سازی داده‌ها 6](#_Toc165282318)

[1-2-1-رسم هیستوگرام 6](#_Toc165282319)

[2-2-1-رسم و تفسیر نمودارهای جعبه‌ای 7](#_Toc165282320)

[3-2-1-رسم و تفسیر ماتریس همبستگی 8](#_Toc165282321)

[3-1-تشخیص داده‌های پرت 9](#_Toc165282322)

[4-1-پر کردن مقادیر خالی 10](#_Toc165282323)

[1-4-1-تعریف شی پیش‌پردازش برای پر کردن مقادیر خالی با Iterative Imputer 10](#_Toc165282324)

[2-4-1-تعریف شی پیش‌پردازش برای پر کردن مقادیر خالی با KNN 13](#_Toc165282325)

[5-1-انتخاب ویژگی 13](#_Toc165282326)

[6-1- آموزش مدل درخت تصمیم 14](#_Toc165282327)

[1-6-1-آموزش مدل با پارامترهای پیشفرض 14](#_Toc165282328)

[2-6-1-پیدا کردن هایپرپارامترهای بهینه با استفاده از GridSearch 15](#_Toc165282329)

[3-6-1-هرس کردن درخت و نمایش مجدد آن (بخش امتیازی) 17](#_Toc165282330)

[2-پاسخ سوال دوم 19](#_Toc165282331)

[1-2-آماده سازی داده‌ها 19](#_Toc165282332)

[2-2-حذف داده‌های پرت 21](#_Toc165282333)

[3-2-انتخاب ویژگی 22](#_Toc165282334)

[4-2-آموزش و تست رگرسیون لجستیک 23](#_Toc165282335)

[3-پاسخ سوال سوم 25](#_Toc165282336)

[1-3-پیش پردازش داده‌ها 27](#_Toc165282337)

[1-1-3-حذف تگ‌های <br /><br /> 27](#_Toc165282338)

[2-1-3-ساخت تابع پیش‌پردازش 27](#_Toc165282339)

[2-3-استفاده از CountVectorizer 28](#_Toc165282340)

[3-3- استفاده از TF-IDF 31](#_Toc165282341)

[4-3-ساخت تابعی که به نظرات جدید برچسب می‌دهد 31](#_Toc165282342)

[4-پاسخ سوال چهارم 33](#_Toc165282343)

[1-4-وارد کردن و پیش‌پردازش داده‌ها 33](#_Toc165282344)

[2-4-آموزش رگرسیون لجستیک چند کلاسه 35](#_Toc165282345)

فهرست شکل‌ها

[شکل 1: وارد کردن کتابخانه‌های مورد نیاز 1](#_Toc165282346)

[شکل 2: خواندن دیتاست 1](#_Toc165282347)

[شکل 3: حذف ستون load\_id 2](#_Toc165282348)

[شکل 4: کد مربوط به نمایش ده سطر تصادفی 2](#_Toc165282349)

[شکل 5: ده سطر تصادفی نمایش داده شده 2](#_Toc165282350)

[شکل 6: کد مربوط به نشان دادن تعداد مقادیر گمشده 2](#_Toc165282351)

[شکل 7: تعداد مقادیر خالی در هر سطر 3](#_Toc165282352)

[شکل 8: کد بررسی تعداد سطرهای تکراری 3](#_Toc165282353)

[شکل 9: کد بررسی تعداد صفرها در هر ستون 3](#_Toc165282354)

[شکل 10: تعداد مقادیر صفر در هر سطر 4](#_Toc165282355)

[شکل 11: بررسی تعداد و جمعیت موجود در سطرهای گسسته 4](#_Toc165282356)

[شکل 12: خروجی کد شکل 11 4](#_Toc165282357)

[شکل 13: خروجی اتریبیوت df.columns 5](#_Toc165282358)

[شکل 14: کد حذف فاصله از ستون‌ها 5](#_Toc165282359)

[شکل 15: جایگذاری صفر با np.nan 5](#_Toc165282360)

[شکل 16: جداسازی داده‌های عددی و دسته‌ای در دو لیست جدا 6](#_Toc165282361)

[شکل 17: کد مربوط به رسم هیستوگرام 6](#_Toc165282362)

[شکل 18: هیستوگرام‌های رسم شده 7](#_Toc165282363)

[شکل 19: کد رسم نمودار جعبه‌ای 7](#_Toc165282364)

[شکل 20: نمودارهای جعبه‌ای 8](#_Toc165282365)

[شکل 21: کد رسم ماتریس همبستگی 8](#_Toc165282366)

[شکل 22: ماتریس همبستگی رسم شده 9](#_Toc165282367)

[شکل 23: حذف داده‌های پرت با روش IQR 10](#_Toc165282368)

[شکل 24: ایجاد شی از کلاس پایپ‌لاین 10](#_Toc165282369)

[شکل 25: ایجاد شی از کلاس ColumnTransformer 11](#_Toc165282370)

[شکل 26: پیش پردازش داده‌های آموزش و تست 11](#_Toc165282371)

[شکل 27: Oversample کردن داده‌های آموزشی 12](#_Toc165282372)

[شکل 28: آموزش و تست مدل رندوم فارست 12](#_Toc165282373)

[شکل 29: محاسبه دقت 12](#_Toc165282374)

[شکل 30: پایپ‌لاین به روز شده 13](#_Toc165282375)

[شکل 31: پر کردن مقادیر خالی با KNN 13](#_Toc165282376)

[شکل 32: انتخاب ویژگی‌ها با استفاده از SelectKBest() 14](#_Toc165282377)

[شکل 33: پایپ‌لاین آموزش مدل پیشفرض 14](#_Toc165282378)

[شکل 34: گزارش به دست آمده از آموزش مدل پیشفرض 15](#_Toc165282379)

[شکل 35: استانداردسازی داده‌ها 15](#_Toc165282380)

[شکل 36: دیکشنری هایپرپارامترها 15](#_Toc165282381)

[شکل 37: پیداکردن ترکیب بهینه هایپرپارامترها با گرید سرچ 16](#_Toc165282382)

[شکل 38: نمایش پارامترهای بهینه 16](#_Toc165282383)

[شکل 39: درخت تصمیم با هایپرپارامترهای بهینه 16](#_Toc165282384)

[شکل 40: گزارش دسته بندی درخت تصمیم با هایپرپارامترهای بهینه 17](#_Toc165282385)

[شکل 41: کد مربوط به نمایش شکل درخت 17](#_Toc165282386)

[شکل 42: درخت به دست آمده از هایپرپارامترهای بهینه‌شده 18](#_Toc165282387)

[شکل 43: گزارش عملکرد درخت هرس شده 18](#_Toc165282388)

[شکل 44: درخت هرس شده 18](#_Toc165282389)

[شکل 45: وارد کردن کتابخانه‌های مورد نیاز 19](#_Toc165282390)

[شکل 46: خروجی متد .describe() 19](#_Toc165282391)

[شکل 47: ده سطر رندوم دیتافریم 20](#_Toc165282392)

[شکل 48: خروجی تعداد هر دسته در هر ستون 20](#_Toc165282393)

[شکل 49: پیدا کردن ؟ در دیتاست 21](#_Toc165282394)

[شکل 50: انکود کردن ستون‌های دودویی 21](#_Toc165282395)

[شکل 51: one-hot-encode کردن داده‌ها 22](#_Toc165282396)

[شکل 52: پیداکردن داده‌های پرت با استفاده از خوشه‌بندی 22](#_Toc165282397)

[شکل 53: پایپ‌لاین ساخته شده برای تحلیل مولفه اصلی 23](#_Toc165282398)

[شکل 54: گزارش به دست آمده از آموزش و تست مدل با 15 مولفه اصلی 23](#_Toc165282399)

[شکل 55: انتخاب 15 مولفه اصلی به عنوان ویژگی‌ها 24](#_Toc165282400)

[شکل 56: ساخت، آموزش و تست مدل رگرسیون لجستیک 24](#_Toc165282401)

[شکل 57: گزارش عملکرد رگرسیون لجستیک 24](#_Toc165282402)

[شکل 58: آموزش مدل با Cross Validation 25](#_Toc165282403)

[شکل 59: نتیجه Cross Validation 25](#_Toc165282404)

[شکل 60: وارد کردن کتابخانه‌های مورد نیاز 25](#_Toc165282405)

[شکل 61: تعداد ایمیل‌های با لیبل مثبت و منفی 26](#_Toc165282406)

[شکل 62: نمایش ده سطر رندوم 26](#_Toc165282407)

[شکل 63: جایگزینی لیبل‌ها با صفر و یک 26](#_Toc165282408)

[شکل 64: حذف تگ‌های <br /><br /> 27](#_Toc165282409)

[شکل 65: ساخت شی ریشه‌یاب و مجموعه ایست‌واژه‌ها 27](#_Toc165282410)

[شکل 66: تابع پیش‌پردازش متن 28](#_Toc165282411)

[شکل 67: پیش‌پردازش متن‌ها با استفاده از تابع تعریف شده 28](#_Toc165282412)

[شکل 68: بردار کردن متون با استفاده از CountVectorizer 28](#_Toc165282413)

[شکل 69: انتخاب 5000 کلمه مهم با استفاده از آزمون مربع کای 29](#_Toc165282414)

[شکل 70: استاندارد سازی و تبدیل ماتریس به ماتریس چگال 29](#_Toc165282415)

[شکل 71: پیش پردازش داده‌های تست 29](#_Toc165282416)

[شکل 72: تبدیل ماتریس‌های چگال به آرایه‌های خلوت 30](#_Toc165282417)

[شکل 73: آموزش و تست مدل بیز 30](#_Toc165282418)

[شکل 74: گزارش عملکرد مدل نایو بیز 30](#_Toc165282419)

[شکل 75: گزارش عملکرد مدل به دست آمده از دیتاست TF-IDF 31](#_Toc165282420)

[شکل 76: ذخیره کردن مدل برای استفاده در آینده 31](#_Toc165282421)

[شکل 77: تابع سنجش عواطف نظرات جدید 32](#_Toc165282422)

[شکل 78: کاربرد تابع تحلیل عواطف 32](#_Toc165282423)

[شکل 79: خروجی تابع تحلیل عواطف 32](#_Toc165282424)

[شکل 80: وارد کردن کتابخانه‌های مورد نیاز 33](#_Toc165282425)

[شکل 81: فایل digits.png 33](#_Toc165282426)

[شکل 82: تابع تقسیم کننده عکس بزرگ به عکس‌های کوچکتر 34](#_Toc165282427)

[شکل 83: آماده سازی داده‌ها برای آموزش 34](#_Toc165282428)

[شکل 84: آموزش و تست رگرسیون لجستیک 35](#_Toc165282429)

[شکل 85: محاسبه دقت 35](#_Toc165282430)

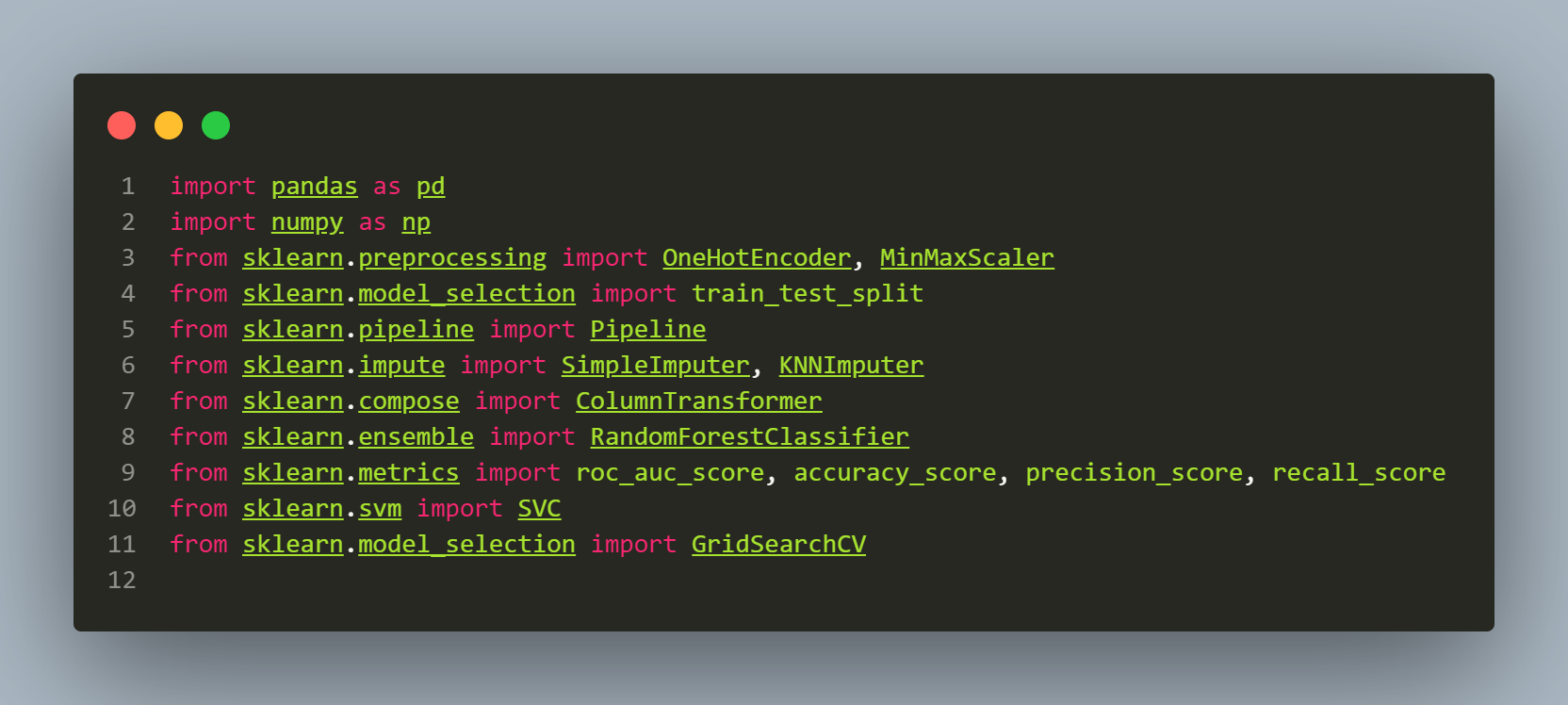
[شکل 86: نحوه محاسبه ماتریس درهم‌ریختگی 36](#_Toc165282431)

[شکل 87: ماتریس درهم‌ریختگی به دست آمده 36](#_Toc165282432)

# 2-پاسخ سوال دوم

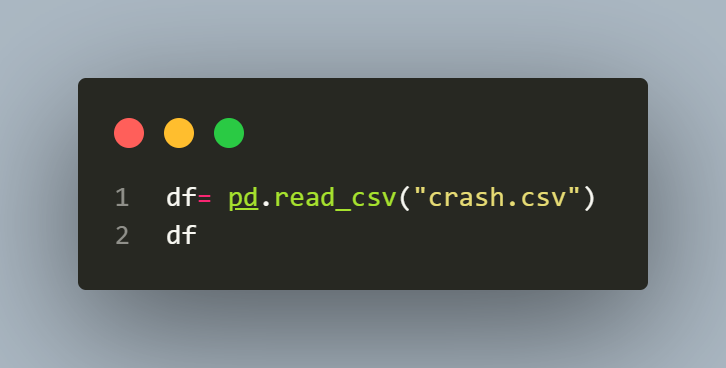
## 1-2-آماده‌سازی داده‌ها

کد شکل (1)، کتابخانه‌های مورد نیاز برای پردازش و تحلیل داده‌ها، از جمله pandas، numpy و ابزارهای مختلف از scikit-learn را وارد می‌کند.



شکل 1: وارد کردن کتابخانه‌های مورد نیاز

داده‌ها از فایل crash.csv بارگذاری شده و در DataFrame به نام df ذخیره می‌شوند. (شکل 2)



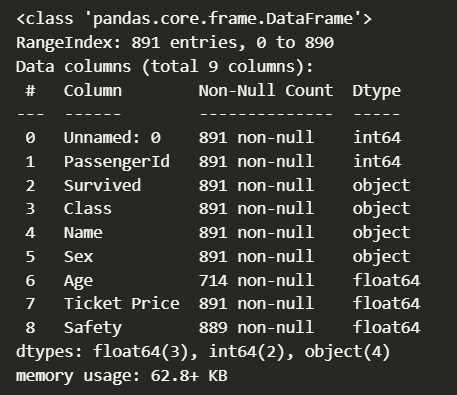
شکل 2: بارگذاری داده‌ها

سپس ساختار داده‌ها و اطلاعات اولیه آن با استفاده از ()df.info نمایش داده می‌شود تا نوع و تعداد مقادیر گمشده بررسی شود. (شکل 3)



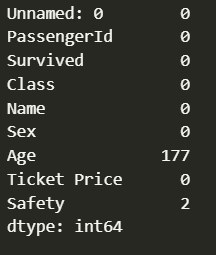
شکل 3: بررسی ساختار اولیه داده‌ها

شکل (4) خروجی این متد را نمایش می‌دهد.



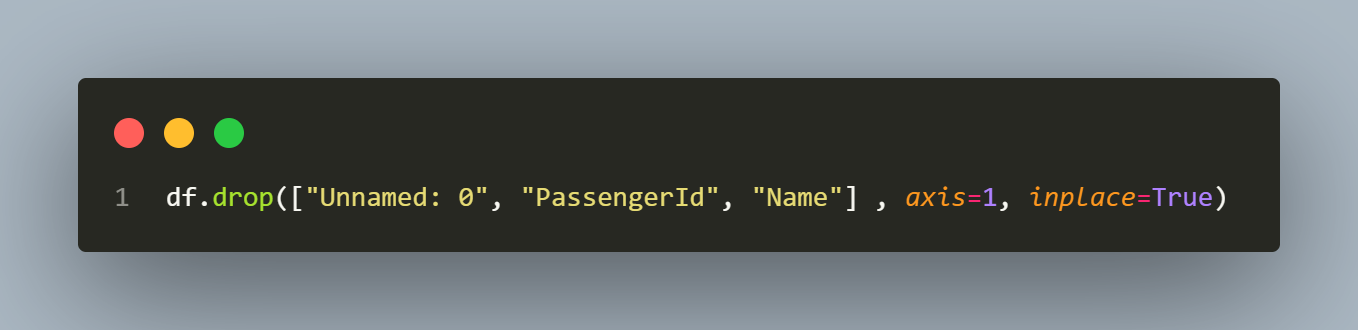
شکل 4: خروجی متد .info()

سپس تعداد مقادیر گمشده در هر ستون داده‌ها با استفاده از ()df.isna().sum محاسبه می‌شود. شکل (5) خروجی این متد را نمایش می‌دهد.



شکل 5: تعداد مقادیر گمشده هر ستون

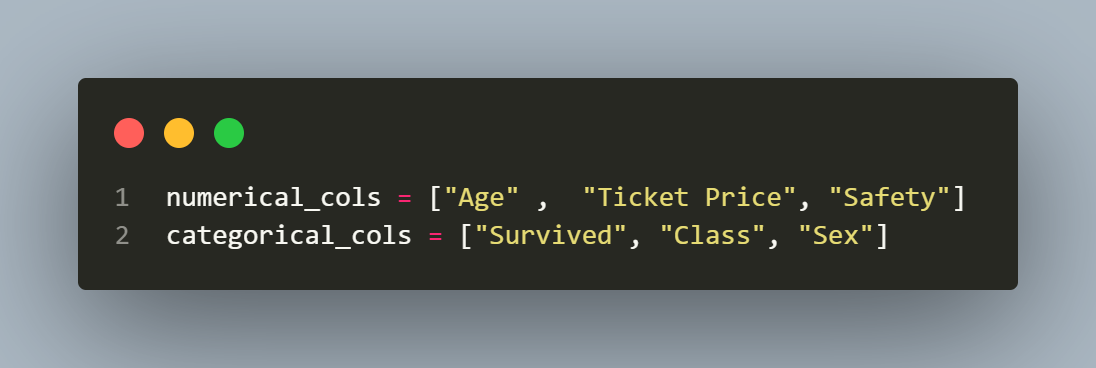
ستون‌های غیر ضروری مانند "Unnamed: 0""، "PassengerId" و "Name" از دیتافریم حذف می‌شوند تا فقط ستون‌های مهم باقی بمانند. این کار با متد .drop() انجام می‌شود. شکل (6) نحوه انجام این کار را نمایش می‌دهد.



شکل 6: حذف ستون‌های غیر ضروری

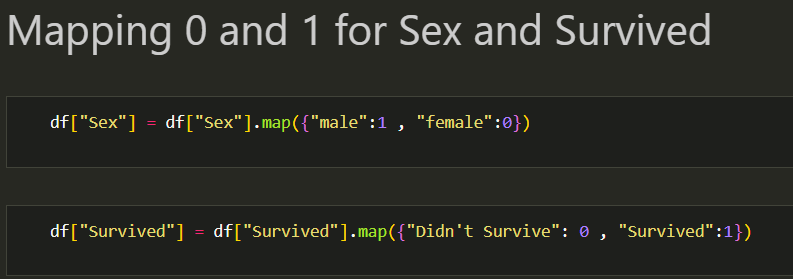
سپس توزیع مقادیر مختلف در ستون‌های "Class" "Survived" و "Sex" با استفاده از value\_counts بررسی می‌شود. در ادامه تعداد مقادیر صفر و "؟" در هر ستون محاسبه و نمایش داده می‌شود. مشاهده می‌شود که 15 مقدار ستون ticke price برابر صفر است. ممکن است این رکوردها یا گم شده باشند (بلیط رایگان کمی غیر منطقی است) یا مربوط به خدمه‌ی هواپیما باشند. در هر صورت، برای اطمینان از یکدست بودن نتایج، این رکوردها را حذف می‌کنیم. همچنین مشاهده می‌شود که هیچ علامت سوالی در دیتاست موجود نمی‌باشد.

در ادامه، ستون‌های عددی و دسته‌ای به صورت جداگانه شناسایی و دسته‌بندی می‌شوند. شکل (7) این ستون‌ها را نمایش می‌دهد.



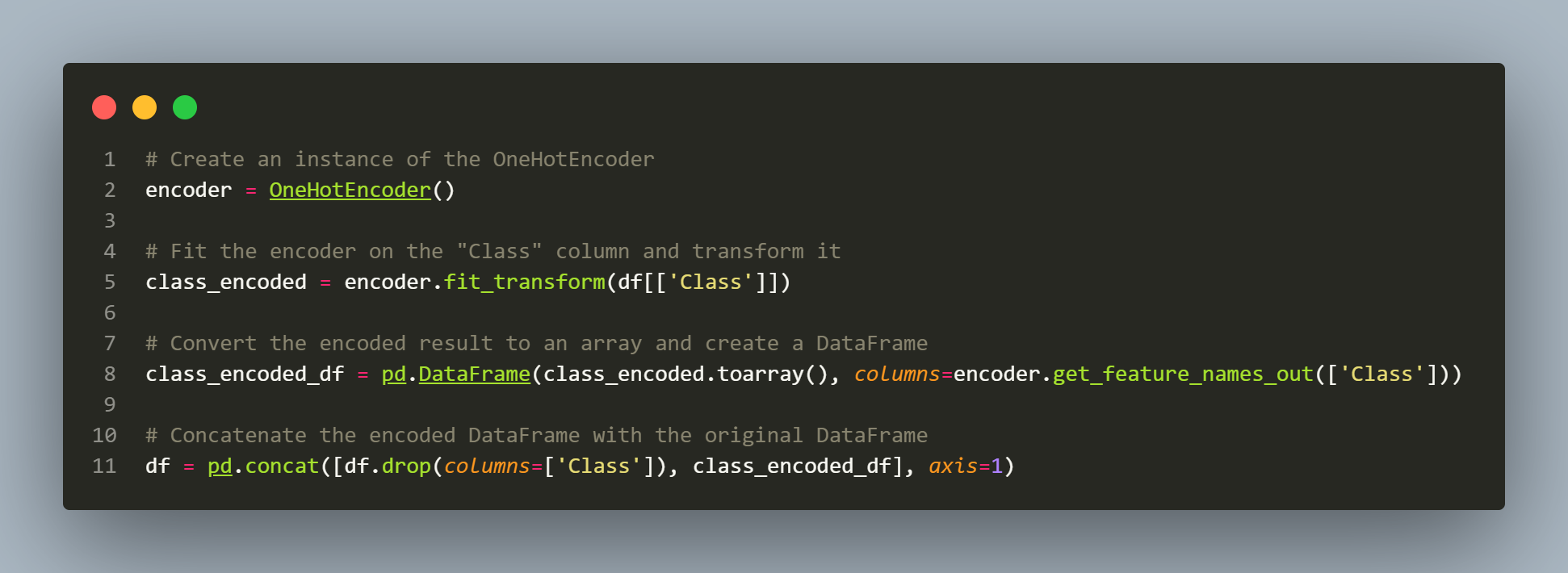
شکل 7: دسته بندی ستون‌های عددی و دسته‌ای

در ادامه مقادیر دسته‌ای در ستون‌های "Sex" و "Survived" به مقادیر عددی تبدیل می‌شوند تا برای مدل‌سازی مناسب باشند. شکل (8) نحوه انجام این کار را با متد .map() نمایش می‌دهد.



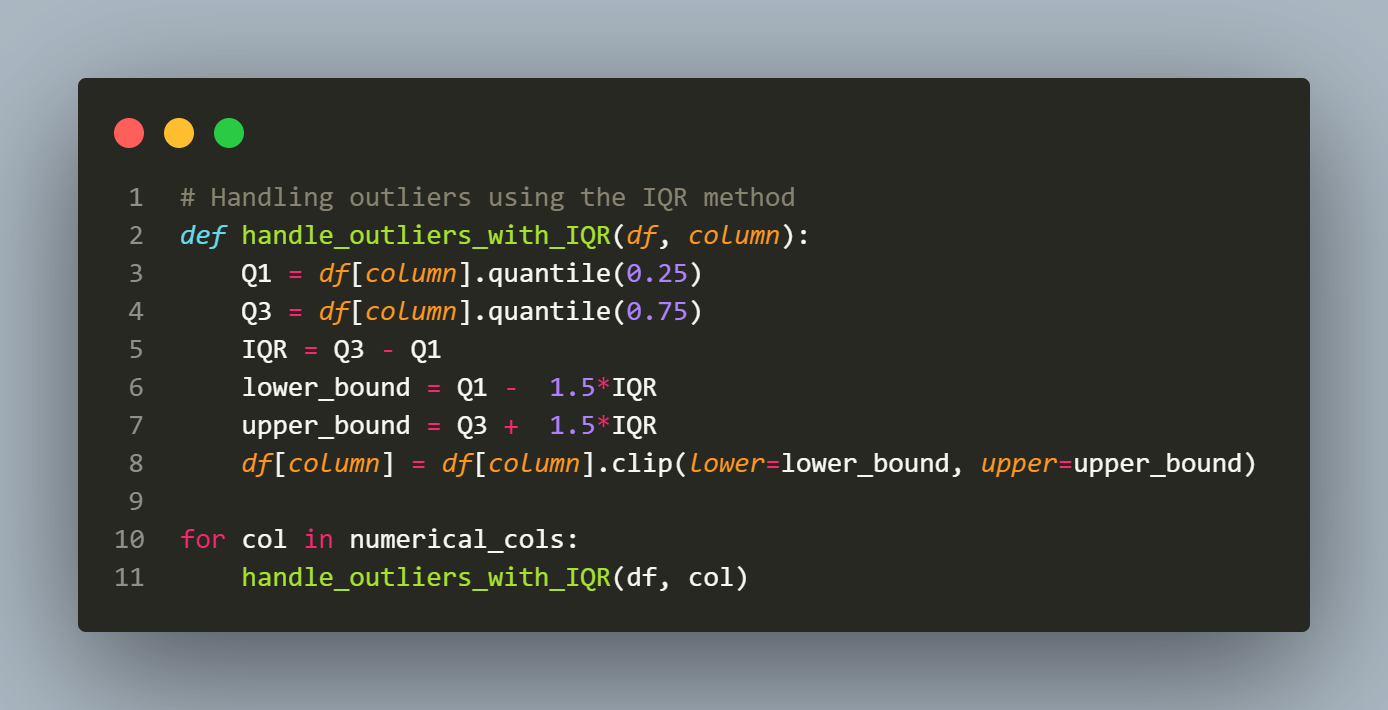
شکل 8: نگاشت مقادیر دسته‌ای در ستون‌های Sex و Survived

سپس مقادیر دسته‌ای در ستون "Class" با استفاده از OneHotEncoder کدگذاری شده و به دیتافریم اصلی اضافه می‌شوند. البته می‌توانستیم از کدگذاری سلسله‌مراتبی هم برای کد گذاری کلاس‌های پرواز استفاده کنیم. اما از آنجایی که کاردینالیتی این ستون برابر 3 است، و تعداد ستون‌ها در مجموع کم است، اضافه کردن دو ستون دیگر به ستون‌های دیتافریم مشکلی ایجاد نمی‌کند. شکل (9) نحوه انجام این کار را نمایش می‌دهد.



شکل 9: OneHotEncoding

در ادامه یک تابع برای مدیریت مقادیر پرت با استفاده از روش IQR (Interquartile Range) تعریف و برای ستون‌های عددی اعمال می‌شود. شکل (10) این تابع را نمایش می‌دهد.



شکل 10: حذف داده‌های پرت با استفاده از روش دامنه میان چارکی

با بررسی اتریبیوت .shape متوجه می‌شویم که هیچ مقداری حذف نشده است. درنتیجه داده پرتی در دیتاست وجود ندارد.

## 2-2-مدیریت مقادیر گمشده

حال به مدیریت مقادیر گمشده می‌پردازیم. مقادیر گمشده در ستون‌های مشخص با استفاده از تکنیک‌های مختلف جایگزین شدند: برای این منظور از دو روش استفاده شد:

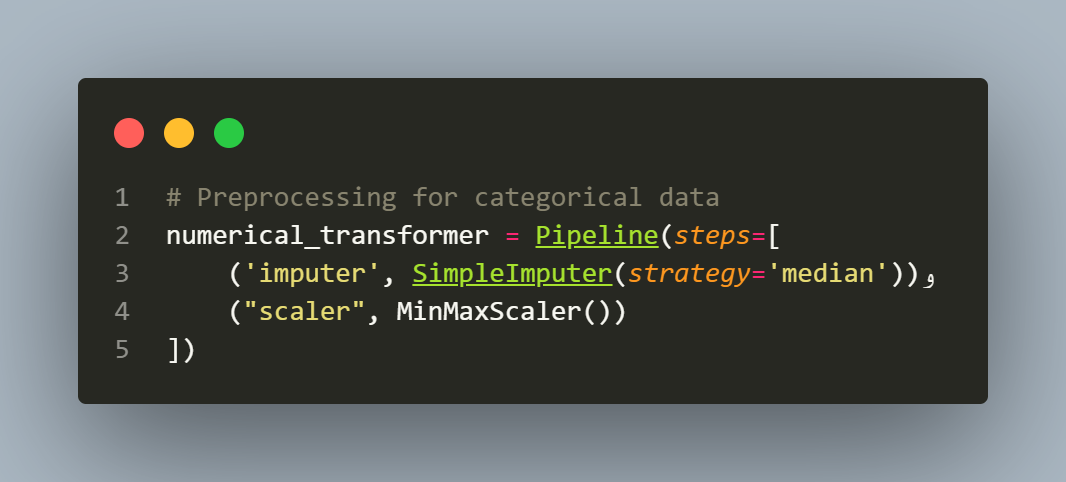
* استفاده از SimpleImputer برای جایگزینی مقادیر گمشده با میانه.
* استفاده از KNNImputer برای جایگزینی مقادیر گمشده با استفاده از مقادیر مشابه در نزدیک‌ترین همسایگان.

حال برای راحتی اجرای پیش پردازش، از کلاس Pipeline از کتابخانه scikitlearn استفاده می‌کنیم. این کلاس، مجموعه‌ای از اعمال را به شکل پیاپی بر روی دیتاست اجرا می‌کند.

پایپ‌لاین مورد استفاده به دو قسمت تقسیم می‌شود: پیش‌پردازش داده‌های عددی، آموزش مدل و تست و ارزیابی مدل.

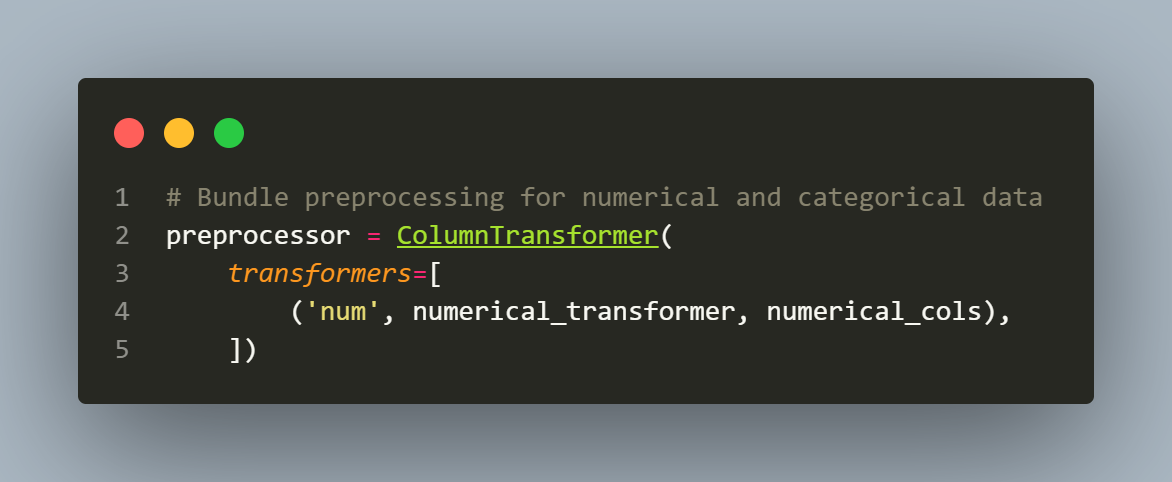
پیش‌پردازش داده‌های دسته‌ای خود شامل یک پایپ‌لاین کوچک‌تر است که شامل اجرای دو column transformer است. ترتیب وارد کردن ترنسفورمرها در شیِ پایپ‌لاین، نماینده ترتیب اجرا نیز می‌باشد. اولین ترنسفورمر SimpleImputer است که کار جایگزینی np.nan را با میانه به عهده دارد. دومین ترنسفورمر روی هر ستون MinMaxScaler را اجرا می‌کند.

نحوه کارکرد این ترنسفورمرها به این صورت است که یک به یک ستون‌‌ها را دریافت می‌کنند، مقادیر خالی را با میانه جایگذاری می‌کند و سپس ستون را با روش مین-مکس مقیاس می‌کند. شکل (11) پایپ‌لاین مربوط به پیش‌پردازش داده‌های عددی را نمایش می‌دهد.



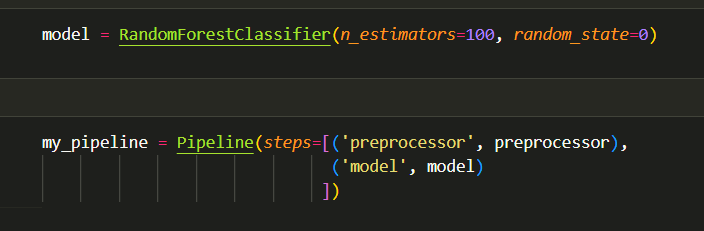
شکل 11: پایپ‌لاین پیش پردازش داده‌های عددی

حال با استفاده از کلاس ColumnTransformer() یک شی به نام preprocessor ایجاد می‌کنیم. در این کلاس، لیستی از تاپل‌ها را به عنوان آرگومان پاس می‌دهیم به طوری که عضو اول تاپل، نام ترنسفورمر، عضو دوم عمل یا اعمالی است که باید روی ستون‌ها انجام شود و عضو سوم لیستی شامل نام ستون‌های هدف است که کارهای تعریف شده بر روی آنها اعمال می‌شود. شکل (12) نحوه تعریف کلاس ColumnTransformer() را نمایش می‌دهد.



شکل 12: نحوه تعریف کلاس ColumnTransformer()

*در قدم بعدی، مدل را تعریف می‌کنیم و پایپ‌لاین اصلی را ایجاد می‌کنیم که شامل دو مرحله‌ی پیش‌پردازش و آموزش است. شکل (13) این بخش را نمایش می‌دهد.*

**

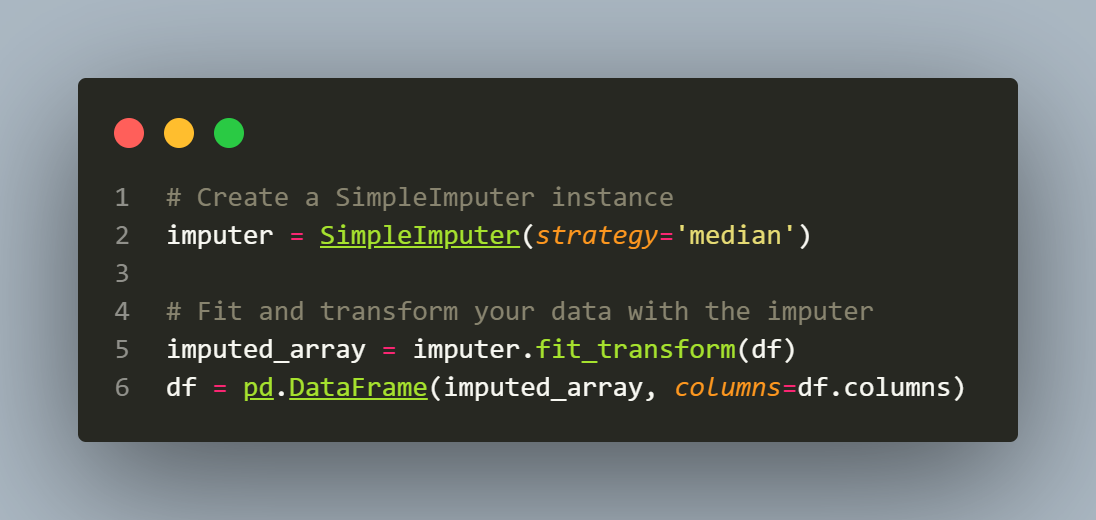
شکل 13: تعریف مدل و پایپ‌لاین اصلی

از آنجایی که این مدل صرفاً نقش یک معیار برای ارزیابی عملکرد روش‌ها را دارد، احتیاجی به fine tune کردن آن نیست چون که این مدل قرار نیست در ادامه کار پیش‌بینی داده‌های جدید را انجام دهد.

در قدم بعدی با استفاده از متد fit() و با پاس دادن داده‌های آموزشی به عنوان آرگومان، ابتدا داده‌ها را طبق توضیحات گفته شده پیش‌پردازش می‌کنیم و سپس مدل را آموزش می‌دهیم. در ادامه با استفاده از متد predict و پاس دادن داده‌های معیارسنجی، ابتدا این داده‌ها پیش‌پردازش می‌شوند و سپس مدل آموزش داده شده روی داده‌های آموزشی بر روی این داده‌ها معیارسنجی می‌شود. مشاهده می‌شود استفاده از میانه مقدار AUC را برابر با 0.6363 نتیجه می‌دهد.

مشابه همین کار را مجدداً تکرار می‌کنیم، با این تفاوت که به جای SimpleImputer از KNNImputer استفاده می‌کنیم. در این حالت AUC برابر 0.6315 به دست می‌آید.

مشاهده می‌شود میانه عملکرد بهتری دارد. پس مقادیر گمشده را با میانه جایگذاری می‌کنیم. شکل (13) نحوه انجام این کار را نمایش می‌دهد.

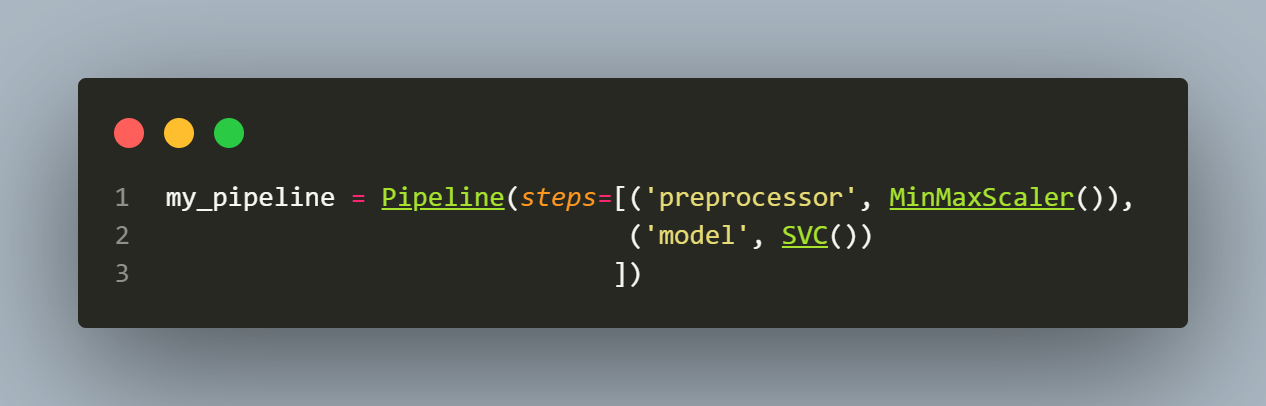


شکل 14: جایگذاری مقادیر خالی با میانه

## 3-2-آموزش مدل ماشین بردار پشتیبان

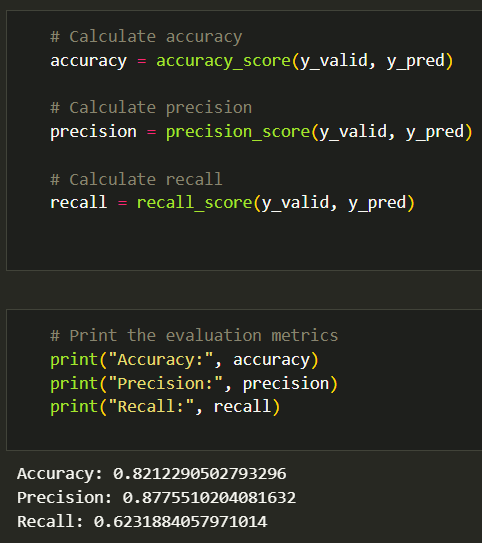
ابتدا داده‌ها به مجموعه‌های آموزشی و ارزیابی تقسیم می‌شوند تا مدل بتواند با داده‌های آموزشی آموزش ببیند و با داده‌های ارزیابی تست شود.

سپس مشابه قبل، یک پایپ‌لاین برای آموزش مدل ایجاد می‌شود. شکل(15) پایپ‌لاین ایجاد شده را نمایش می‌دهد.



شکل 15: پایپ‌لاین آموزش SVM

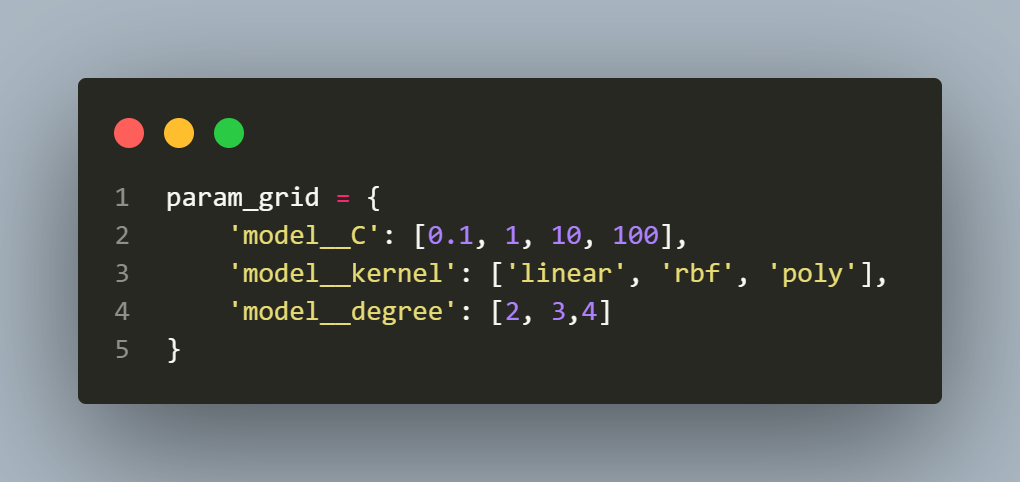
سپس مشابه بخش قبلی، مدل را آموزش می‌دهیم و تست می‌کنیم و معیارهای ارزیابی را محاسبه می‌کنیم. شکل (16) محاسبه معیارهای ارزیابی و نتیجه‌ آنها را نمایش می‌دهد.



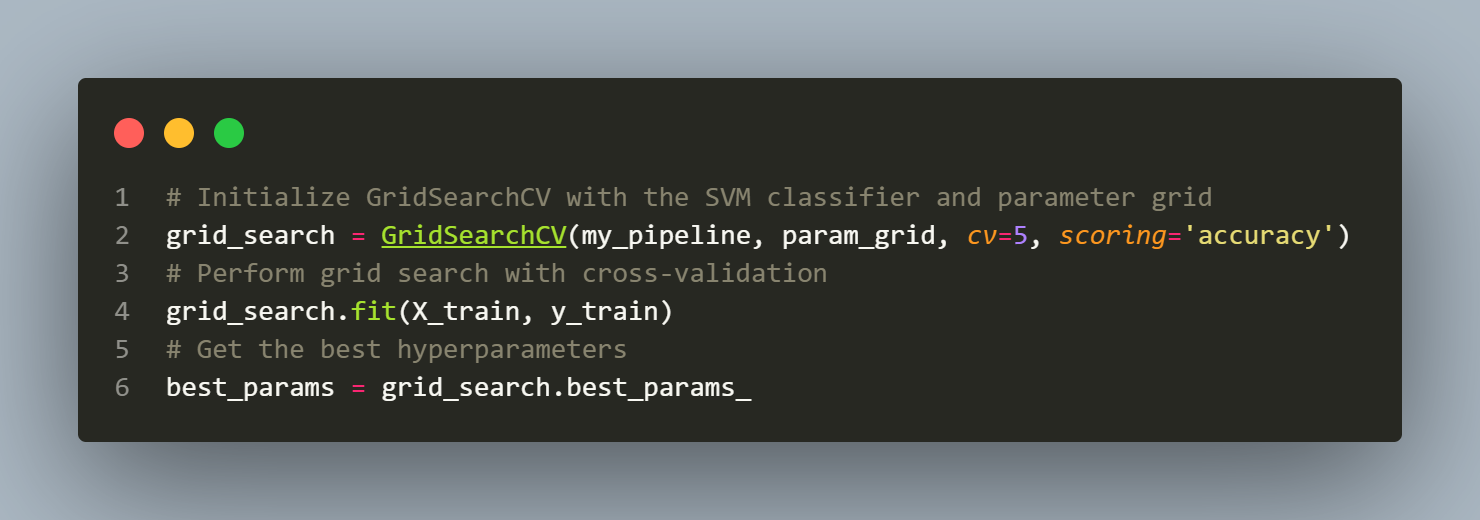
شکل 16: محاسبه و نمایش مقادیر معیارهای ارزیابی

## 4-2-بهینه‌سازی هایپرپارامترها

شبکه‌ای از پارامترها برای جستجوی شبکه‌ای ایجاد می‌شود و یک پایپ‌لاین جدید، با استفاده از GridSearchCV بهینه‌سازی می‌شود. شکل (17) پارامترها و شکل (18) جست و جوی شبکه‌ای را نمایش می‌دهد.

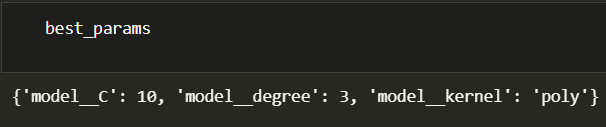


شکل 17: پارامترهای مدل



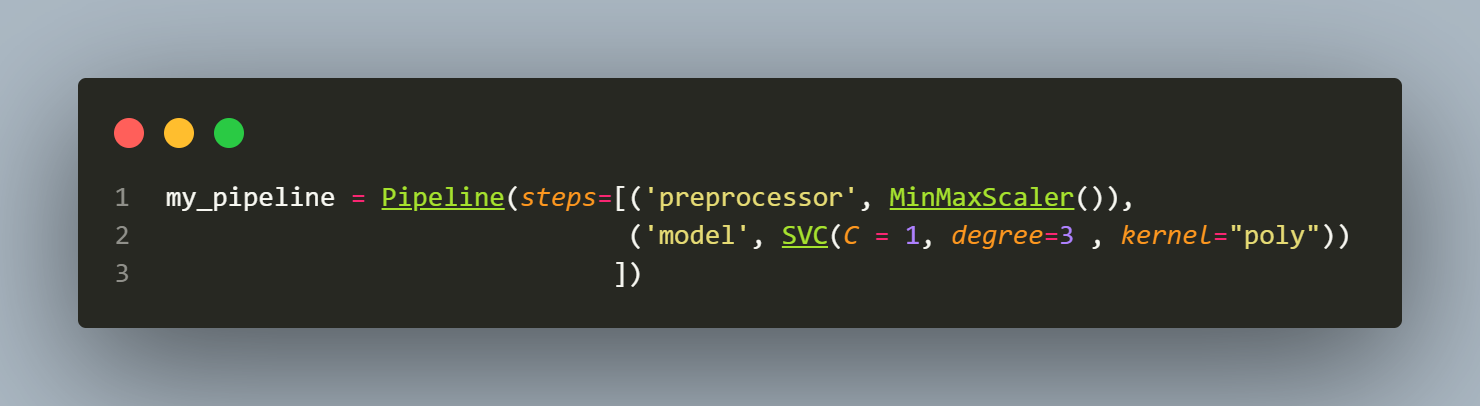
شکل 18: انجام جست و جوی شبکه‌ای

سپس مقادیر بهینه را نمایش می‌دهیم. شکل (19) هایپرپارامترهای بهینه را نمایش می‌دهد.

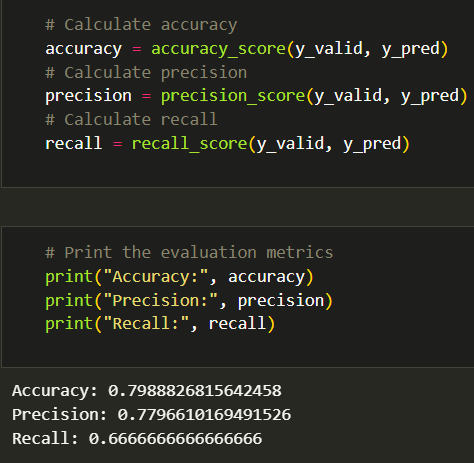


شکل 19: هایپرپارامترهای بهینه

سپس مشابه بخش‌های قبلی با استفاده از یک پایپ‌لاین مدل با پارامترهای بهینه را آموزش می‌دهیم. پایپ‌لاین به روز شده در شکل (20) نمایش داده شده است و نتایج مربوطه در شکل (21) قرار گرفته.



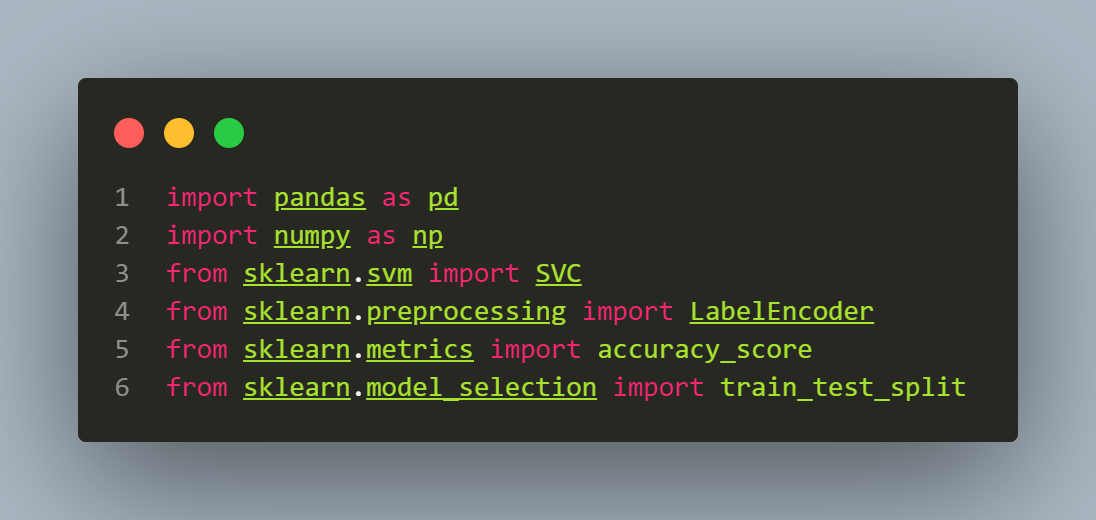
شکل 20: پایپ‌لاین با هایپرپارامترهای بهینه



شکل 21: نتایج بهینه‌سازی پارامترها

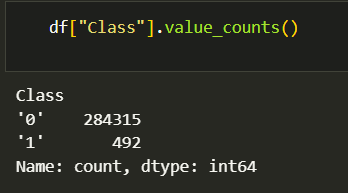
# 3-پاسخ سوال سوم

کتابخانه‌های مورد نیاز برای تحلیل داده‌ها و مدل‌سازی وارد می‌شوند. این شامل pandas، numpy و ابزارهای مختلف از scikit-learn برای پیش‌پردازش داده‌ها، تقسیم داده‌ها، ساخت مدل‌ها و ارزیابی مدل‌ها است. شکل (22) این کتابخانه‌ها را نمایش می‌دهد.



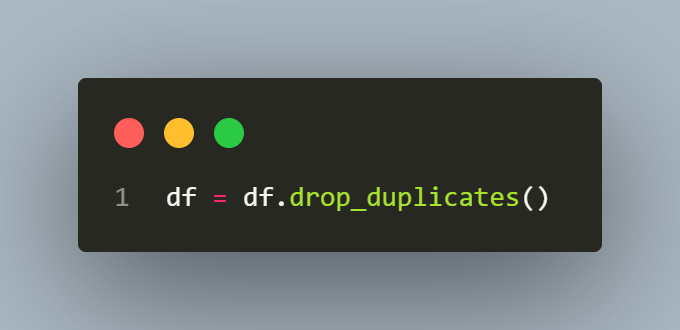
شکل 22: ایمپورت کردن کتابخانه‌های مورد نیاز

پس از فراخوانی دیتاست، با استفاده از متد .values\_counts() تعداد هر یک از کلاس‌ها را بررسی می‌کنیم. شکل (23) خروجی این متد را نمایش می‌دهد.



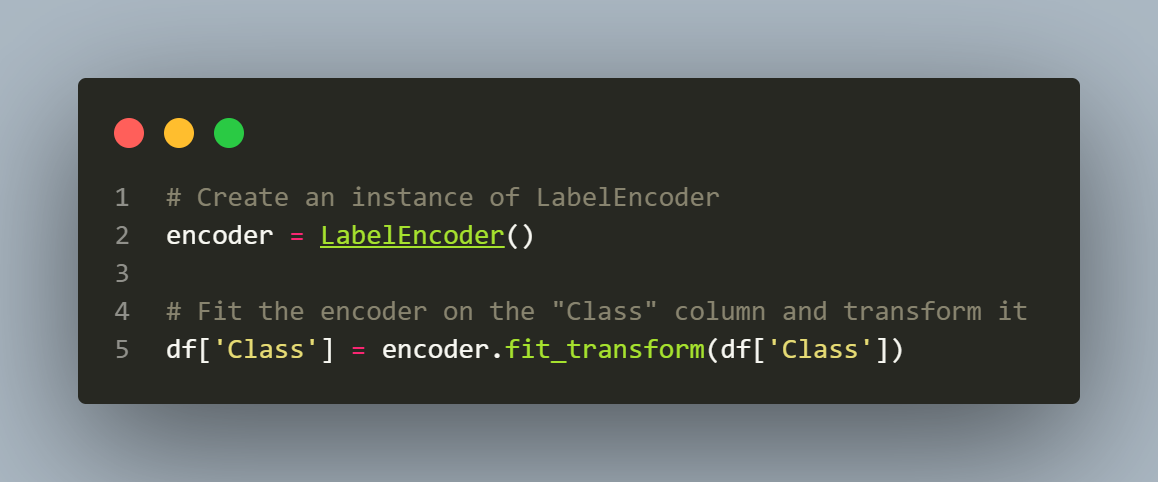
شکل 23: بررسی تعداد هر یک از کلاس‌ها

سپس داده‌های تکراری شناسایی و حذف می‌شوند تا کیفیت داده‌ها بهبود یابد. شکل (24) نحوه انجام این کار را نمایش می‌دهد.



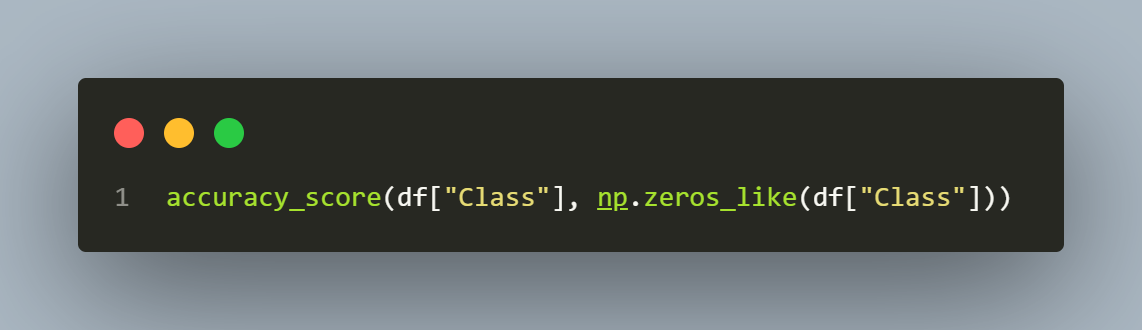
شکل 24: حذف ردیف‌های تکراری

سپس ستون "Time" را با متد .drop() حذف می‌کنیم. در ادامه با استفاده از .labelEncoder() ستون هدف را به صفر و یک انکود می‌کنیم. شکل (25) نحوه انجام این کار را نمایش می‌دهد.



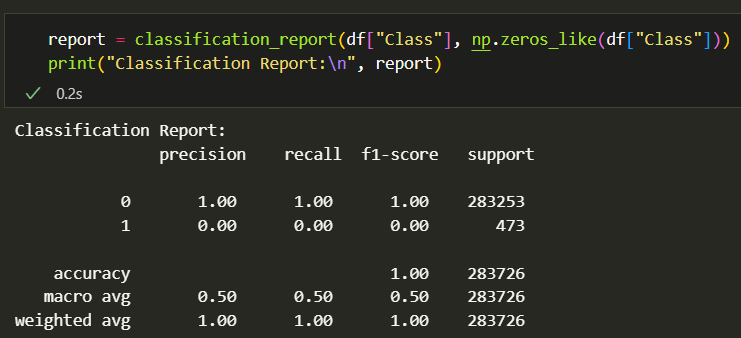
شکل 25: انکود کردن ستون هدف

در مرحله بعد، مدل اولیه‌ای با استفاده از آرایه‌ای شامل صفر برای ارزیابی دقت ایجاد می‌شود. این مرحله به عنوان پایه‌ای برای مقایسه با مدل‌های پیشرفته‌تر استفاده می‌شود.



شکل 26: ارزیابی دقت با آرایه‌ای متشکل از صفر

دقت پایه برابر 0.9983 است. همچنین یک گزارش عملکرد از این حال پایه می‌گیریم. شکل (27) این گزارش را نمایش می‌أهد.



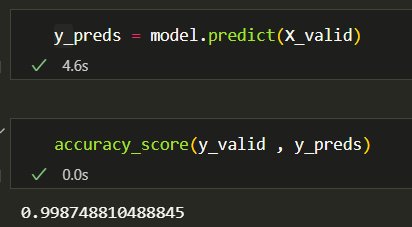
شکل 27:گزارش عملکرد در حالت پایه

پس از تقسیم کردن دیتاست به دیتاست آموزشی و آزمایشی، برای آموزش بهتر مدل، باید تعداد کلاس‌ها را در دیتاست بالانس کنیم. برای این منظور، از under sampling بهره می‌گیریم. روش مورد نظر برای انجام این کار Tomek links است. Tomek Links جفت‌هایی از نمونه‌ها را از کلاس‌های مختلف شناسایی می‌کند که نزدیک‌ترین همسایه‌های یکدیگر هستند. حذف نمونه های کلاس اکثریت در این جفت‌ها به تمیز کردن مرزهای کلاس کمک می کند. سپس مدل را روی این دیتاست بالانس شده آموزش می‌دهیم. شکل (28) نحوه تقسیم‌بندی، بالانس کردن دیتا با کلاس TomekLinks() و آموزش مدل را نمایش می‌دهد.



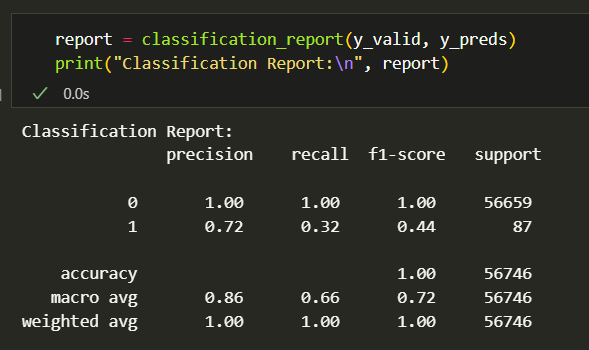
شکل 28: آموزش مدل با داده‌های بالانس شده

سپس مدل را تست می‌کنیم و دقت را ارزیابی می‌کنیم. شکل (29) تست مدل و خروجی دقت را نمایش می‌دهد.



شکل 29: تست مدل و ارزیابی دقت

همچنین شکل (30) گزارش عملکرد مدل را نمایش می‌دهد.



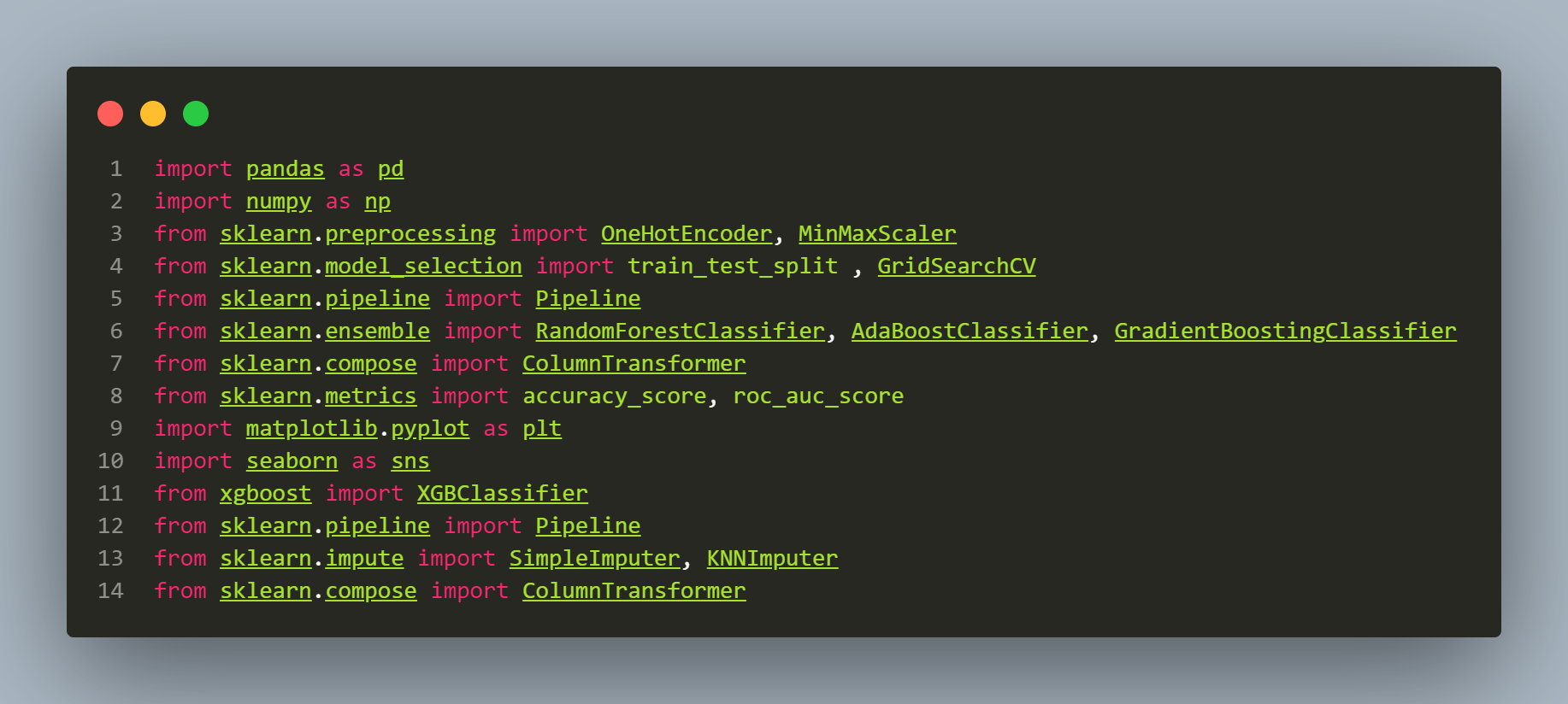
شکل 30: گزارش عملکرد مدل آموزش داده شده

مشاهده می‌شود که مدل آموزش داده شده، عملکرد بهتری نسبت به حالت پایه دارد.

در یک مجموعه داده نامتعادل، افزایش دقت برای کلاس اقلیت (نامعتبر) اغلب شامل چندین استراتژی است که می تواند به بهبود عملکرد مدل کمک کند. این روش‌ها می‌توانند بر دقت کلی و هر کلاس تأثیر بگذارند و در تقاضاهای محاسباتی متفاوت هستند. می‌توان از روش‌های under sampling کمک گرفت. این روش‌ها نیاز به قدرت محاسباتی کمتری دارند، زیرا تعداد داده‌های آموزشی را بسیار کم می‌کنند. روش‌های oversampling نیز موجود هستند که تلاش می‌کنند از کلاس اقلیت، نمونه‌های مصنوعی درست کنند. این روش‌ها احتیاج به پردازش بیشتری دارند زیرا حجم داده‌های آموزشی را بسیار زیاد می‌کنند.

# 4-پاسخ سوال چهارم

شکل (31) کتابخانه‌های مورد نیاز را نمایش می‌دهد.



شکل 31: کتابخانه‌های مورد نیاز

همانند قبل دیتاست را بارگذاری می‌کنیم و تعداد مقادیر گمشده و تعداد صفر در ستون‌ها را بررسی می‌کنیم.