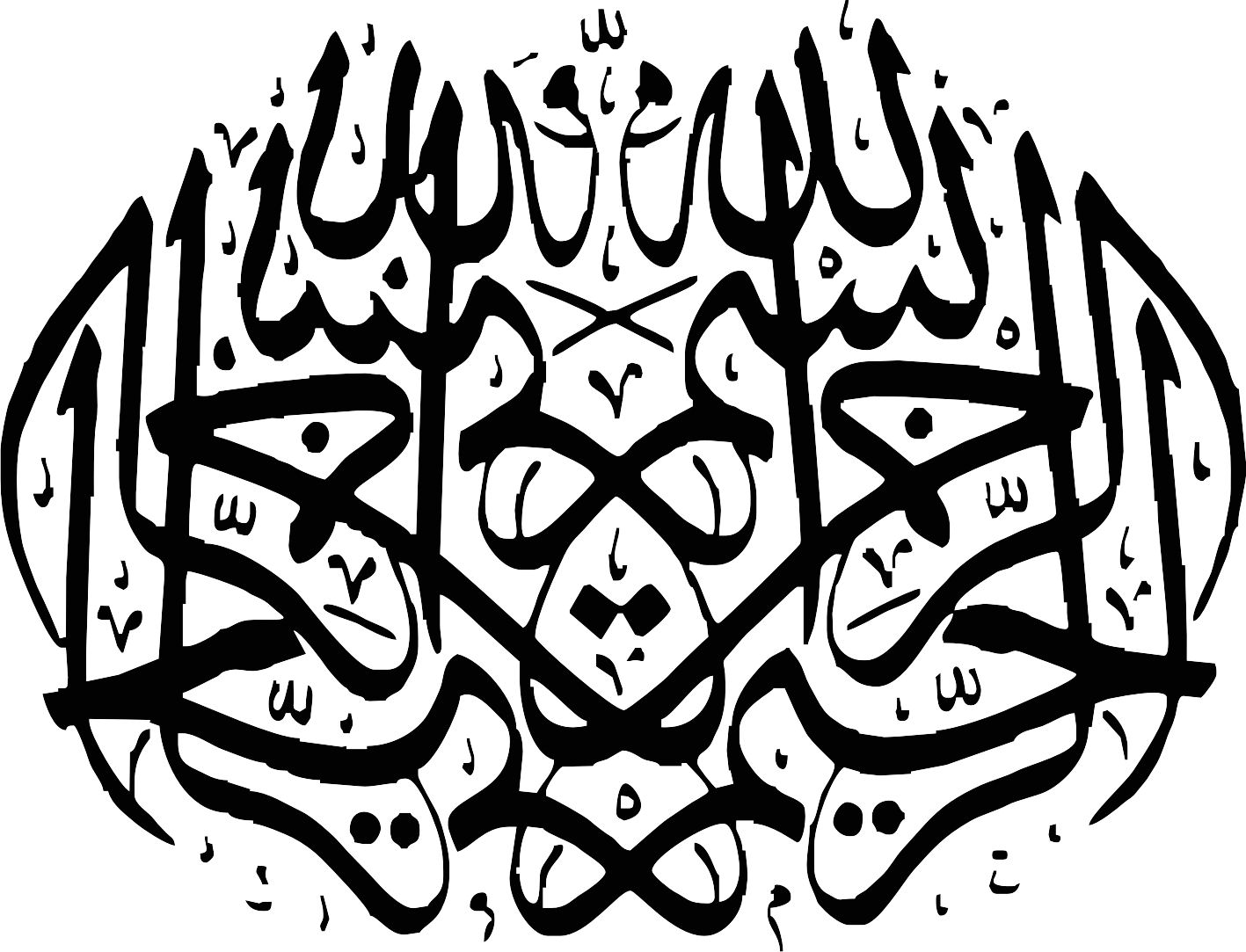


**گزارش تکلیف دوم درس یادگیری ماشین کاربردی**

**استاد درس: دکتر ناظرفرد**

تهیه کننده: سید نیما محمودیان

شماره دانشجویی: 402125005



فهرست مطالب

[1-پاسخ سوال اول 1](#_Toc163208856)

[1-1-لود کردن دیتاست، نمایش ده ردیف به صورت تصادفی، و بررسی کلی داده‌ها 1](#_Toc163208857)

[2-1-مدیریت مقادیر گمشده 5](#_Toc163208858)

[3-1-مصور سازی داده‌ها 8](#_Toc163208859)

[4-1-مدیریت داده‌های پرت 14](#_Toc163208860)

[5-1-مهندسی ویژگی‌ها 16](#_Toc163208861)

[2-پیش‌پردازش تصویر 18](#_Toc163208862)

[1-2- gray scale کردن عکس‌ها 20](#_Toc163208863)

[2-2-تنظیم روشنایی و کنتراست تصاویر 21](#_Toc163208864)

[3-2-نرمال سازی تصاویر 23](#_Toc163208865)

[3-پیش‌پردازش متن 25](#_Toc163208866)

[1-3-خواندن داده‌ها از پیکره همشهری 25](#_Toc163208867)

[2-3-پیش‌پردازش متن‌ها 29](#_Toc163208868)

[3-3-TF-IDF 32](#_Toc163208869)

[4-3-مصورسازی داده‌ها 33](#_Toc163208870)

[4-پیوست 40](#_Toc163208871)

فهرست شکل‌ها

[شکل 1: وارد کردن کتابخانه‌های مورد نیاز 1](#_Toc163166210)

[شکل 2: حذف ستون‌های اضافه 1](#_Toc163166211)

[شکل 3: خواندن دیتاست و نمایش ده سطر تصادفی 2](#_Toc163166212)

[شکل 4: بررسی مجموع مقادیر ناموجود در هر ستون 2](#_Toc163166213)

[شکل 5: اطلاعات نمایش داده شده توسط متد info() 3](#_Toc163166214)

[شکل 6: تعداد مقادیر موجود در هر یک از متغیرهای دسته‌ای 3](#_Toc163166215)

[شکل 7: دسته‌بندی مجدد ستون nationality 4](#_Toc163166216)

[شکل 8: دسته‌بندی مجدد ستون edu 5](#_Toc163166217)

[شکل 9: جایگذاری ؟ با np.nan 6](#_Toc163166218)

[شکل 10: train test split 6](#_Toc163166219)

[شکل 11: پایپ‌لاین پیش پردازش داده‌های دسته‌ای 6](#_Toc163166220)

[شکل 12: نحوه تعریف کلاس ColumnTransformer() 7](#_Toc163166221)

[شکل 13: تعریف مدل و پایپ‌لاین اصلی 7](#_Toc163166222)

[شکل 14: شمای کلی پایپ‌لاین 8](#_Toc163166223)

[شکل 15:معیارسنجی مدل 8](#_Toc163166224)

[شکل 16: حذف سطرهای دارای مقادیر خالی 8](#_Toc163166225)

[شکل 17: کد رسم نمودار ستونی 9](#_Toc163166226)

[شکل 18: کد مربوط به نمودار جعبه‌ای 9](#_Toc163166227)

[شکل 19: نمودارهای ستونی رسم شده از داده‌های دسته‌ای 10](#_Toc163166228)

[شکل 20: نمودار جعبه‌ای از داده‌های عددی 11](#_Toc163166229)

[شکل 21: کد مربوط به هیستوگرام 11](#_Toc163166230)

[شکل 22: هیستوگرام‌های رسم شده 11](#_Toc163166231)

[شکل 23: نحوه انتخاب ستون‌های عددی و محاسبه ماتریس همبستگی 12](#_Toc163166232)

[شکل 24: هیت‌مپ رسم شده 12](#_Toc163166233)

[شکل 25: نحوه محاسبه همبستگی با ستون هدف 13](#_Toc163166234)

[شکل 26: نمودار میله‌ای همبستگی با ستون هدف 13](#_Toc163166235)

[شکل 27: نحوه محاسبه هیت‌مپ برای داده‌های دسته‌ای 14](#_Toc163166236)

[شکل 28: هیت‌مپ ایجاد شده از ستون‌های دسته‌ای 15](#_Toc163166237)

[شکل 29: نحوه حذف سطرهایی که داده پرت دارند 15](#_Toc163166238)

[شکل 30: one hot encoding 16](#_Toc163166239)

[شکل 31: انتقال ستون هدف به انتهای دیتافریم 17](#_Toc163166240)

[شکل 32: پایپ‌لاین ارزیابی روش PCA 17](#_Toc163166241)

[شکل 33: ایجاد کلاس و پایپ‌لاین برای mutual information 18](#_Toc163166242)

[شکل 34: شمای کلی پایپ‌لاین ایجاد شده برای mutual information 18](#_Toc163166243)

[شکل 35: وارد کردن کتابخانه‌ها و تولید سه عدد تصادفی 19](#_Toc163166244)

[شکل 36: لود کردن تصاویر و ذخیره آنها در متغیرها 19](#_Toc163166245)

[شکل 37: نمایش ابعاد عکس‌ها 19](#_Toc163166246)

[شکل 38: تبدیل عکس‌ها به rgb و نمایش آنها 20](#_Toc163166247)

[شکل 39: gary scale کردن عکس‌ها 22](#_Toc163166248)

[شکل 40: تابع تنظیم کنتراست و روشنایی 23](#_Toc163166249)

[شکل 41: تنظیم روشنایی و کنتراست شکل اول 24](#_Toc163166250)

[شکل 42: تابع نرمال کننده عکس 25](#_Toc163166251)

[شکل 43: نرمال کردن عکس با استفاده از تابع 25](#_Toc163166252)

[شکل 44: کتابخانه‌های استفاده شده در سوال سوم 26](#_Toc163166253)

[شکل 45: جداسازی رکوردها و قرار دادن آنها در چهار فیلد 26](#_Toc163166254)

[شکل 46: تابع تقسیم کننده فایل به یازده بخش 27](#_Toc163166255)

[شکل 47: تمیز کردن متن‌ها و ذخیره آنها در اکسل 28](#_Toc163166256)

[شکل 48: تبدیل فایل‌های اکسل به یک دیتافریم 28](#_Toc163166257)

[شکل 49: پیدا کردن نحوه انکودینگ متن 29](#_Toc163166258)

[شکل 50: حذف و جایگذینی کاراکترها و type casting 29](#_Toc163166259)

[شکل 51: حذف علائم نگارشی 30](#_Toc163166260)

[شکل 52: نحوه حذف اعداد 30](#_Toc163166261)

[شکل 53: توکنایز کردن متن 31](#_Toc163166262)

[شکل 54: حذف stop wordsها 31](#_Toc163166263)

[شکل 55: نمایش 5 توکنی که بیشتری استفاده را داشته‌اند 31](#_Toc163166264)

[شکل 56: تبدیل توکن‌ها به متن و نرمال سازی متن 32](#_Toc163166265)

[شکل 57: اجرای TF-IDF 32](#_Toc163166266)

[شکل 58: نمایش 5 کلمه مهم هر متن 33](#_Toc163166267)

[شکل 59: نمایش 5 کلمه مهم در همه متون 33](#_Toc163166268)

[شکل 60: فرایند ایجاد wordcloud 34](#_Toc163166269)

[شکل 61: ابر کلمه تولید شده 35](#_Toc163166270)

[شکل 62: انجام تحلیل عواطف با استفاده از کتابخانه polyglot 36](#_Toc163166271)

[شکل 63: نمودار سری زمانی 37](#_Toc163166272)

[شکل 64: نمودار میله‌ای تعداد متن‌های مثبت، خنثی و منفی 37](#_Toc163166273)

[شکل 65: نمودار سهم هر دسته از قطبیت‌ها از کل متون 38](#_Toc163166274)

[شکل 66: هسیتوگرام میزان قطبیت 38](#_Toc163166275)

[شکل 67: نمودار تعداد اخبار مثبت، خنثی و منفی در طول زمان 39](#_Toc163166276)

[شکل 68: نمودار میله‌ای تعداد متون در بیست دسته اول موضوعات 40](#_Toc163166277)

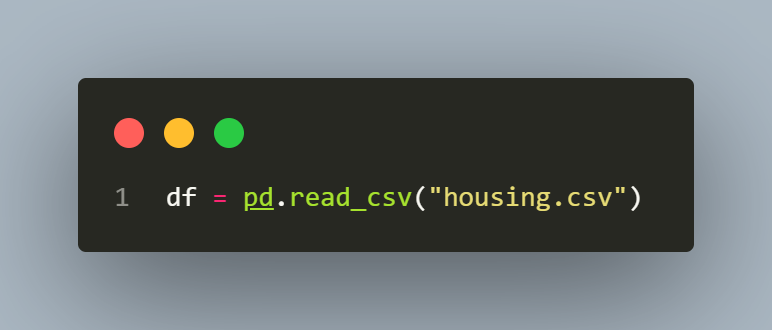
# 1-پاسخ سوال اول

## 1-1-لود کردن دیتاست، نمایش ده ردیف به صورت تصادفی، و بررسی کلی داده‌ها

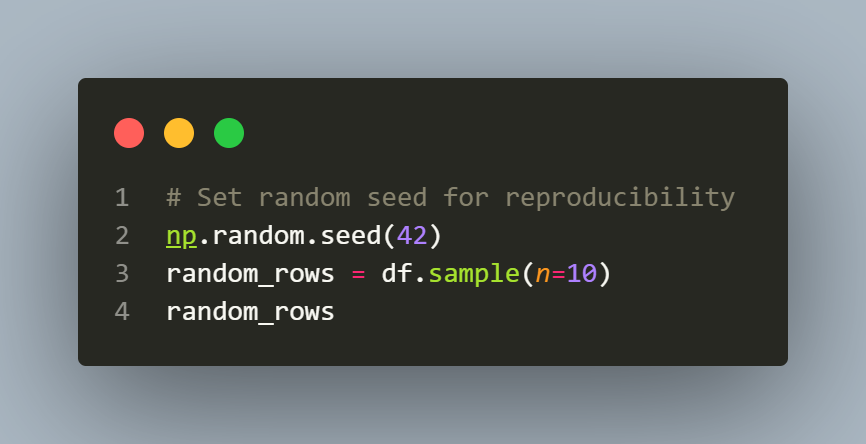
توجه شود که هرجا احتیاجی به ایپمورت کردن یک کتابخانه بود، برای حفظ نظم، ایمپورت در اولین بلوک کد انجام شده است شکل یک بلوک مربوط به وارد کردن کتابخانه‌ها را نمایش می‌دهد. برای لود کردن دیتاست، و نمایش ده ردیف رندوم از کتابخانه pandas استفاده می‌کنیم. با استفاده از کتابخانه numpy یک random seed ایجاد می‌کنیم. Random seed به مظور بازتولید نتایج مشابه تعریف می‌شود. شکل (2)، خواندن دیتاست و شکل (3) نشان دادن ده ردیف رندوم را نمایش می‌دهد.



شکل 1: وارد کردن کتابخانه‌های مورد نیاز



شکل 2: خواندن دیتاست



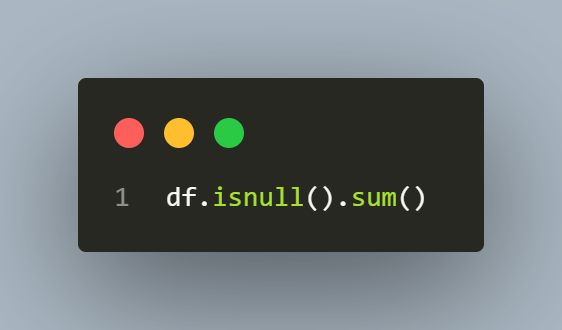
شکل 3: کد مربوط به نمایش ده سطر تصادفی

همچنین شکل (4) ده سطر تصادفی نمایش داده شده را نشان می‌دهد.



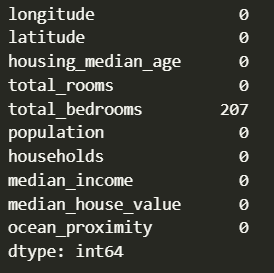
شکل 4: ده سطر تصادفی نمایش داده شده

در ادامه با استفاده از متد info() نگاهی کلی به ستون‌های باقی‌مانده می‌اندازیم. سپس با استفاده از method chaining نمایش داده شده در شکل (5) بررسی می‌کنیم که در هر ستون در مجموع چه تعداد مقادیر تعریف نشده وجود دارد.



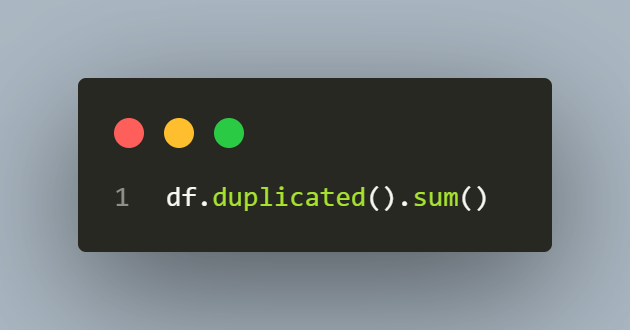
شکل 5: کد مربوط به نشان دادن تعداد مقادیر گمشده

با توجه به شکل (6) مشاهده می‌شود که ستون total\_bedrooms دارای مقادیر خالی است که در ادامه به مدیریت آنها خواهیم پرداخت.

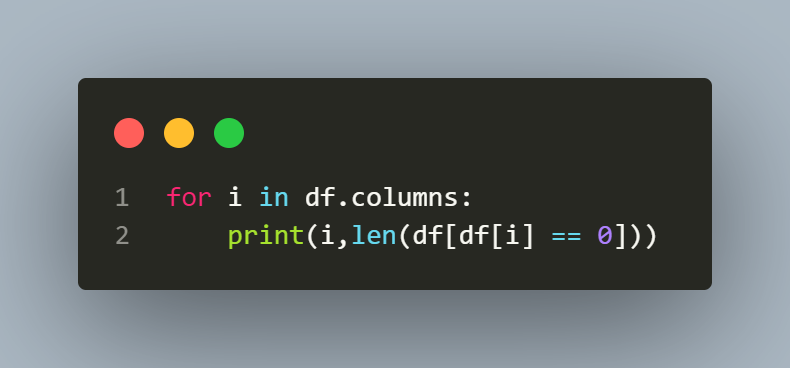


شکل 6: تعداد مقادیر خالی در هر سطر

در ادامه سه مورد را بررسی می‌کنیم: اول اینکه چه تعداد سطر تکراری داریم. شکل (7) نحوه انجام این کار را نمایش می‌دهد. سپس بررسی می‌کنیم که آیا در ستون‌هایی که مقدار صفر معنایی ندارد، رکوردی با مقدار صفر داریم یا خیر. شکل (8) نحوه انجام این کار را نمایش می‌دهد.



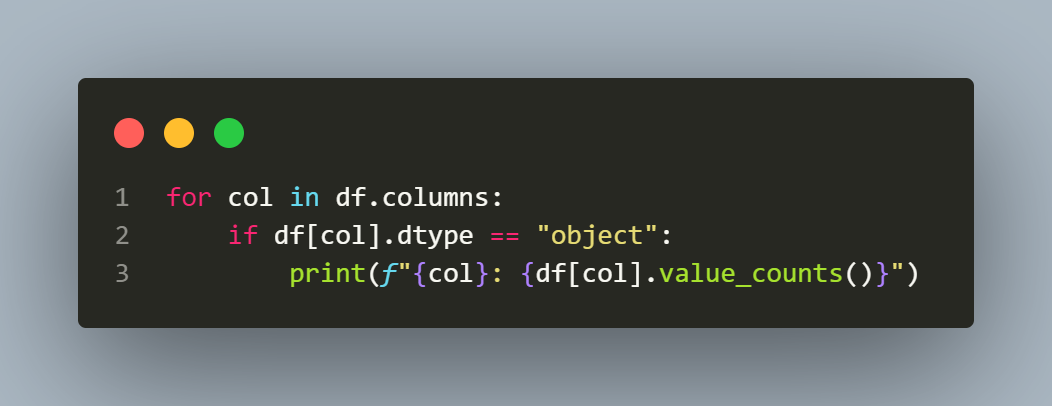
شکل 7: کد بررسی تعداد سطرهای تکراری



شکل 8: کد بررسی تعداد صفرها در هر ستون

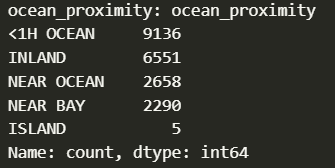
با بررسی خروجی این دو کد، متوجه می‌شویم که هیچ سطر تکراری در دیتافریم وجود ندارد. همچنین در هیچ یک از ستون‌ها مقدار صفر نداریم.

در قدم بعدی بررسی می‌کنیم که در سطرهایی که نوع داده‌ی دسته‌ای دارد، چه کلاس‌هایی موجود داریم و جمعیت هر دسته چقدر است.



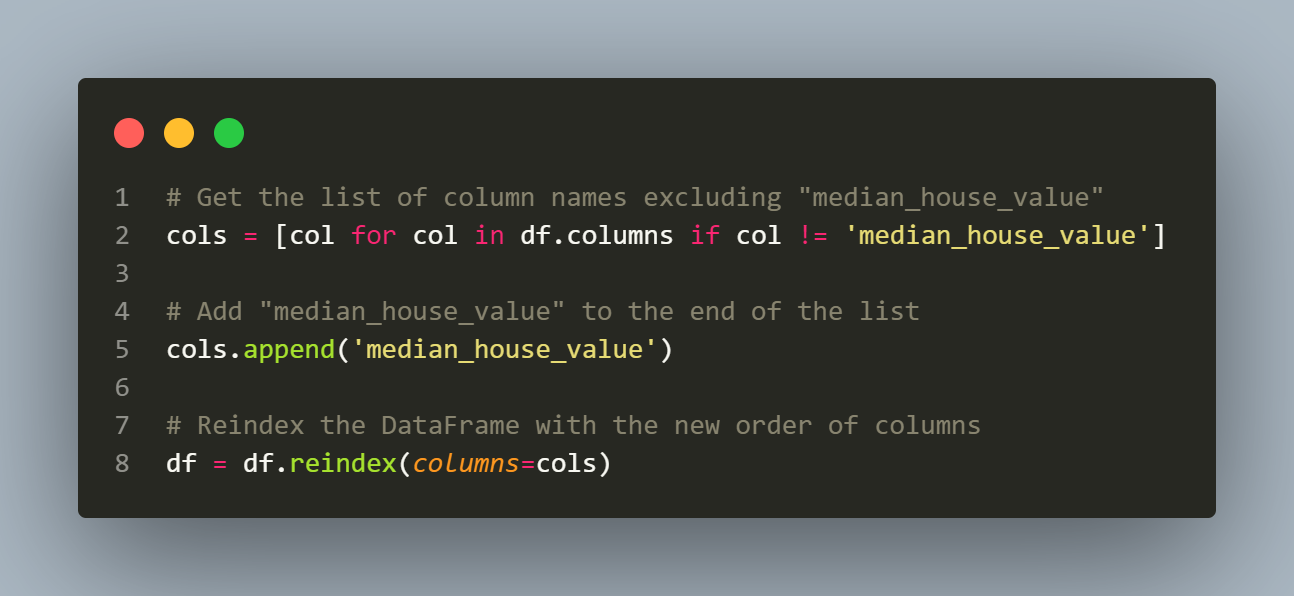
شکل 9: بررسی تعداد و جمعیت موجود در سطرهای گسسته

شکل (10) خروجی این کد را نمایش می‌دهد. مشاهده می‌شود که دسته‌ای با نام غیر معمول در ستون ocean\_proximity وجود ندارد.



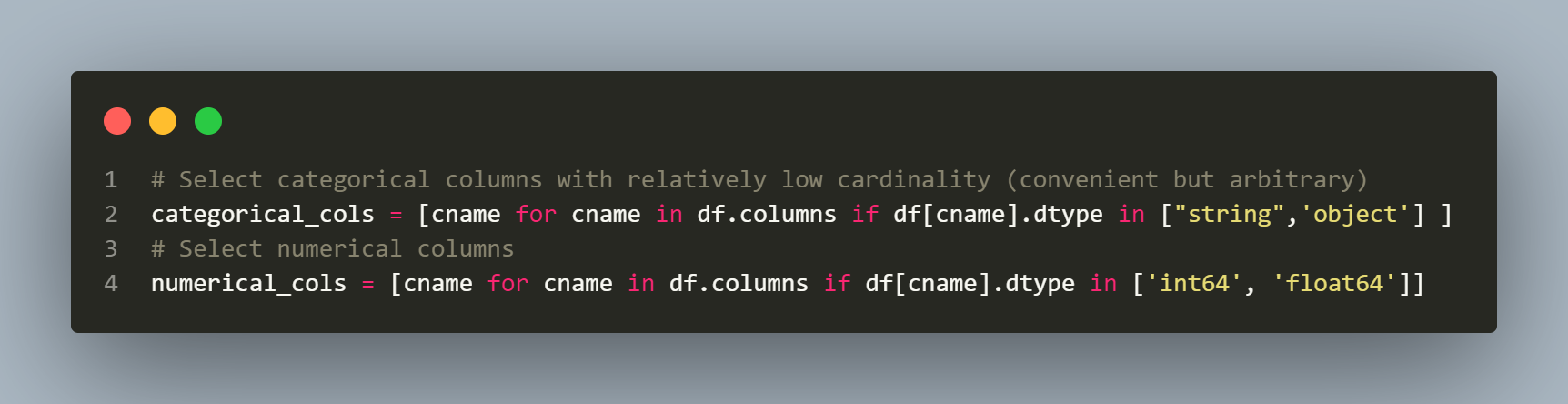
شکل 10: خروجی کد شکل 9

حال برای حفظ قرارداد، سطر هدف را به سطر آخر دیتافریم انتقال می‌دهیم. شکل (11) کدی که این کار را بر عهده دارد نمایش می‌دهد.



شکل 11: انتقال سطر هدف به سطر آخر

حال یک تقسیم‌بندی انجام می‌دهیم که در ادامه‌ی کار مفید واقع می‌شود. با استفاده از List comprehension نام ستون‌هایی که داده‌های عددی دارند را در یک لیست و نام ستون‌هایی که دارای مقادیر دسته‌ای هستند را در یک لیست دیگر قرار می‌دهیم. شکل (12) نحوه انجام این کار را نمایش می‌دهد.



شکل 12: جداسازی داده‌های عددی و دسته‌ای در دو لیست جدا

## 2-1-مصور سازی داده‌ها

### 1-2-1-رسم و تفسیر هیستوگرام

شکل 13 نحوه رسم هیستوگرام را نمایش می‌دهد. num\_rows و num\_cols ابعاد شبکه فرعی را مشخص می کنند. در این مورد، یک شبکه سه در سه ایجاد می کند، بنابراین می توان تا 9 نمودار فرعی را در خود جای داد. تابع plt.subplots برای تولید یک شکل و مجموعه ای از نمودارهای فرعی (محور) استفاده می شود. (20,10) figsize اندازه کل شکل (20 اینچ عرض و 10 اینچ ارتفاع) را مشخص می کند.

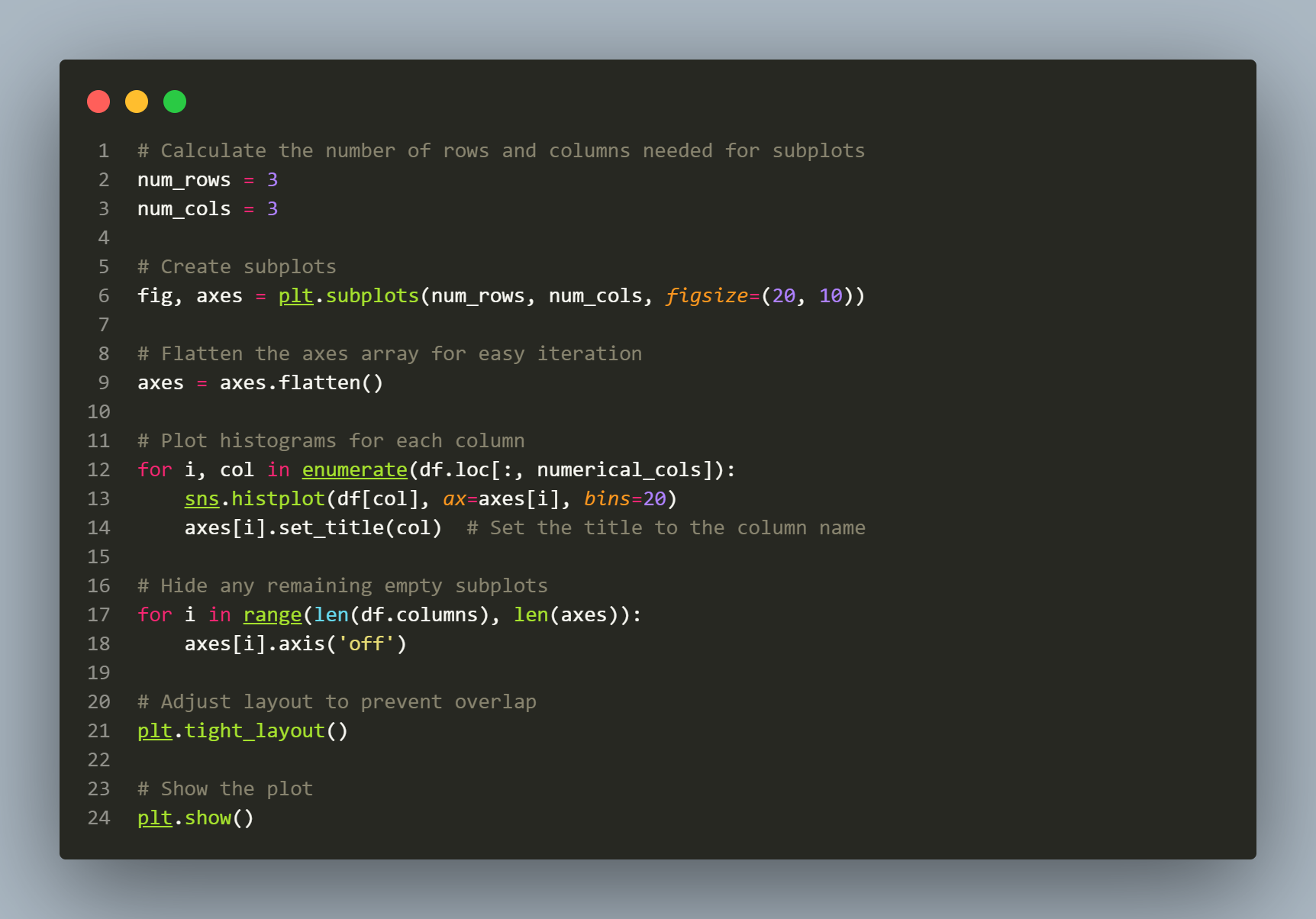
سپس حلقه روی هر ستون مشخص شده در لیست numerical\_cols از df تکرار می شود. برای هر ستون:

sns.histplot یک هیستوگرام از ستون ایجاد می کند و آن را بر روی نمودار فرعی مربوطه ترسیم می کند bins=20 مشخص می کند که هر هیستوگرام باید 20 میله داشته باشد. سپس عنوان هر نمودار بالای آن مشخص می‌شود.

سپس لوپ بعدی هر فضای خالی که در آن نمودار وجود ندارد را حذف می‌کند.

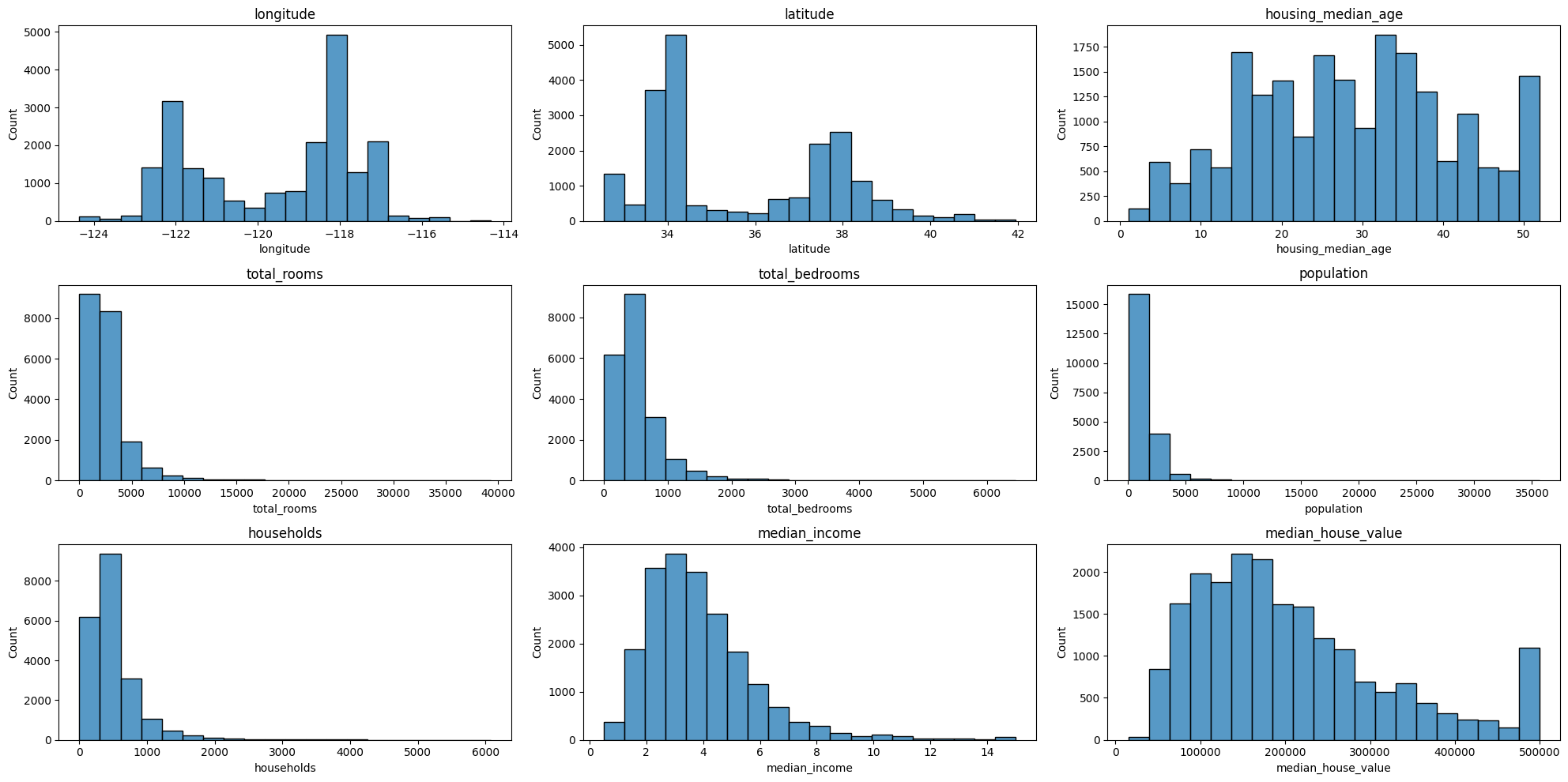
()plt.tight\_layout، نمودارهای فرعی و برچسب ها را طوری تنظیم می کند که به خوبی در ناحیه شکل قرار گیرند و روی هم قرار نگیرند.

plt.show () شکل کامل را با تمام نمودارهای فرعی نمایش می دهد.



شکل 13: کد مربوط به رسم هیستوگرام

همچنین شکل (14) هیستوگرام‌های رسم شده را نمایش می‌دهد.



شکل 14: هیستوگرام‌های رسم شده

**تفسیر:**

**Longitude:**

توزیع چند وجهی است، که نشان دهنده چندین قله است که در آن داده های مسکن به صورت طولی متمرکز شده اند.

این نشان می دهد که خوشه های متعددی از خانه ها در طول های جغرافیایی مختلف، احتمالاً مربوط به شهرها یا مناطق مختلف است.

**Latitude:**

مشابه طول جغرافیایی، توزیع چند وجهی با تمرکز داده های مسکن در عرض های جغرافیایی خاص است. این خوشه ها را در مناطق جغرافیایی خاصی نشان می دهد، که وقتی با طول جغرافیایی ترکیب می شوند، می توانند مناطق جغرافیایی خاصی را مشخص کنند.

**Housing Median Age:**

این هیستوگرام توزیع نسبتاً یکنواختی با قله های متعدد نشان می دهد که نشان می دهد خانه هایی در طیف وسیعی از سنین وجود دارد.

برای خانه های بسیار جدید و بسیار قدیمی، با تعداد خانه های بیشتر در محدوده سنی متوسط، افت جزئی وجود دارد.

**Total Rooms:**

چولگی به سمت راست است، به این معنی که بیشتر نقاط داده در سمت چپ قرار دارند، با تعداد خانه های کمتری که تعداد کل اتاق ها بسیار زیاد است.

تعداد کمی از خانه ها با تعداد بسیار زیاد اتاق وجود دارد که به صورت پرت به نظر می رسند.

**Total Bedrooms:**

مشابه کل اتاق ها، این هیستوگرام نیز دارای انحراف راست است.

اکثر خانه ها اتاق خواب های کمتری دارند و با افزایش تعداد اتاق خواب ها، تعداد آنها کاهش می یابد.

**Population:**

هیستوگرام جمعیت نیز دارای انحراف راست است، که مناطق زیادی با جمعیت کوچکتر و مناطق کمتر با جمعیت بسیار زیاد را نشان می دهد.

**Households:**

این یکی دیگر از توزیع‌های منحرف به راست است که با افزایش تعداد خانوارها، کاهش شدیدی دارد. بیشتر نقاط داده در ناحیه ای متمرکز شده اند که خانوارهای کمتری را نشان می دهد.

**Median Income:**

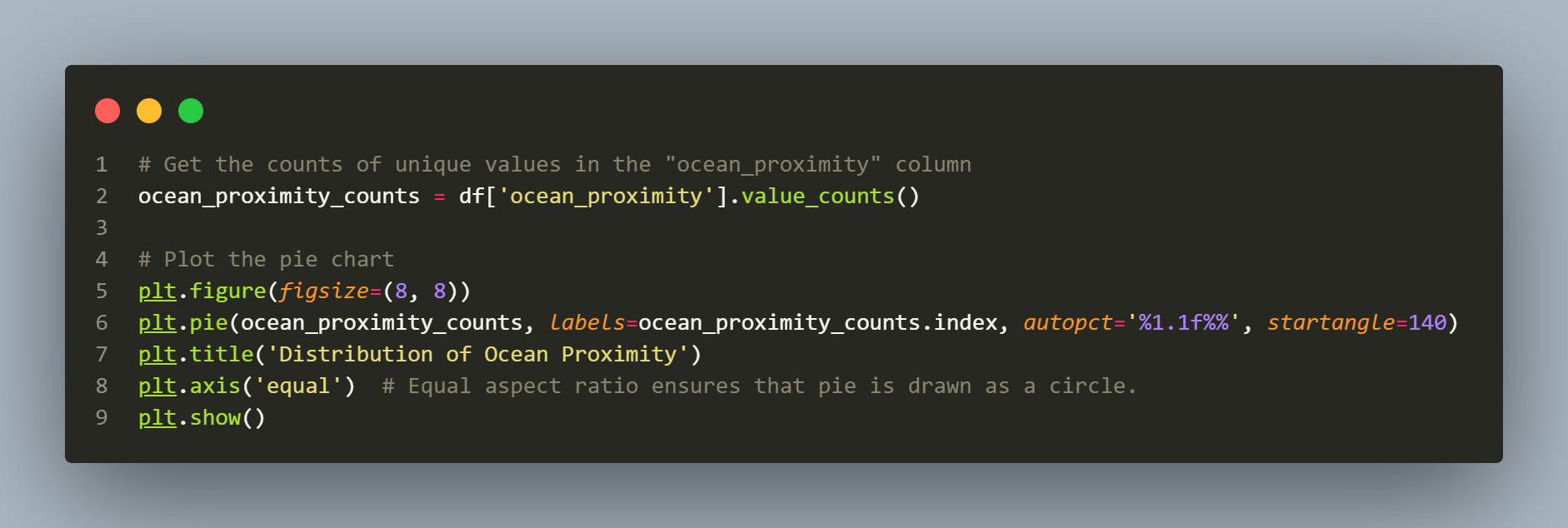
به نظر می‌رسد توزیع میانه درآمد کمی چوله به راست است، اما به طور نرمال‌تر از سایر ویژگی‌ها توزیع می‌شود، که نشان می‌دهد سطوح درآمد به طور یکنواخت‌تر در میان مجموعه داده‌ها توزیع می‌شود، البته با دنباله‌ای به سمت درآمدهای بالاتر.

**Median House Value:**

هیستوگرام مقدار متوسط خانه یک توزیع چند وجهی با چندین قله را نشان می دهد، که نشان می دهد خانه ها در دسته های ارزش مجزا قرار می گیرند. این نشان‌دهنده قیمت‌های متنوع مسکن، با اوج‌ها در محدوده‌های ارزشی خاص است.

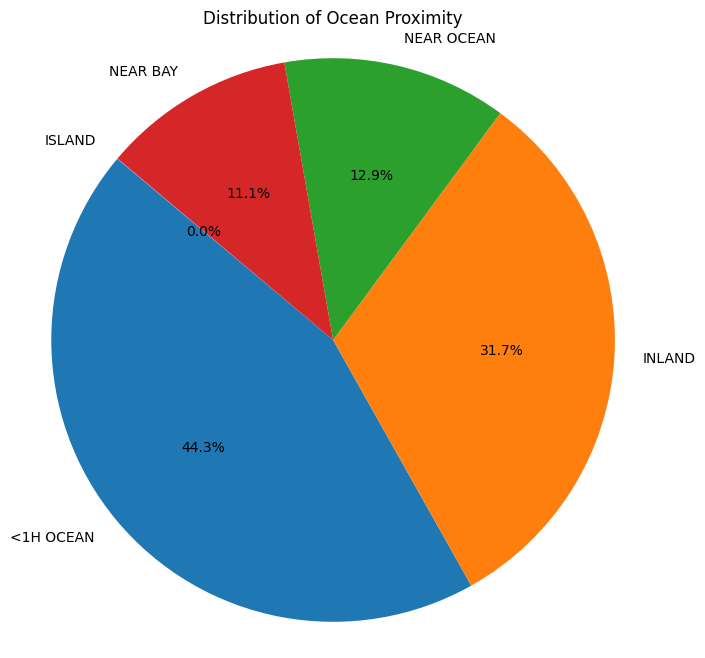
### 2-2-1- رسم نمودار دایره‌ای و تفسیر آن

برای تنها ستون داده‌های دسته‌ای این دیتافریم یک نمودار دایره‌ای رسم می‌کنیم. نحوه رسم این نمودار به این صورت است که ابتدا جمعیت هر دسته را با متد .value\_counts() محاسبه می‌کنیم سپس با استفاده از کتابخانه matplotlib نمودار را رسم می‌کنیم. شکل (15) نمودار رسم شده را نمایش می‌دهد.



شکل 15: کد رسم نمودار دایره‌ای

همچنین شکل (16) خروجی این کد را نمایش می‌دهد.



شکل 16: نمودار دایره‌ای

**تفسیر:**

نزدیکی به اقیانوس: داده‌ها نشان می‌دهند که طبقه‌بندی واضحی از مسکن‌ها بر اساس نزدیکی به اقیانوس وجود دارد و اکثر خانه‌ها به اقیانوس نزدیک هستند.

همچنین بخش قابل توجهی از خانه‌ها از اقیانوس دور هستند که نشان می‌دهد دیتاست پراکندگی خوبی دارد.

خانه‌های ساخته شده در جزیره در این مجموعه داده یا وجود ندارند یا بسیار نادر هستند.

### 3-2-1-رسم و تفسیر نمودارهای جعبه‌ای

رسم نمودارهای جعبه‌ای برای ستونهای عددی همانند رسم هیستوگرام‌ها است با این تفاوت که از تابع boxplot() برای رسم نمودار استفاده می‌شود. شکل (17) کد رسم کننده این نمودارها را نمایش می‌دهد. همچنین شکل (18) خروجی این کد را نمایش می‌دهد.

**تفسیر:**

**Longitude and Latitude:**

این دو نمودار موقعیت جغرافیایی املاک را نشان می دهند. جعبه ها طول کمی دارند و محدوده متمرکزی از مقادیر را با چند نقطه پرت نشان می دهند. این گسترش حاکی از یک منطقه جغرافیایی محدود در حال مطالعه است، مانند یک منطقه یا ایالت خاص.

Housing Median Age:

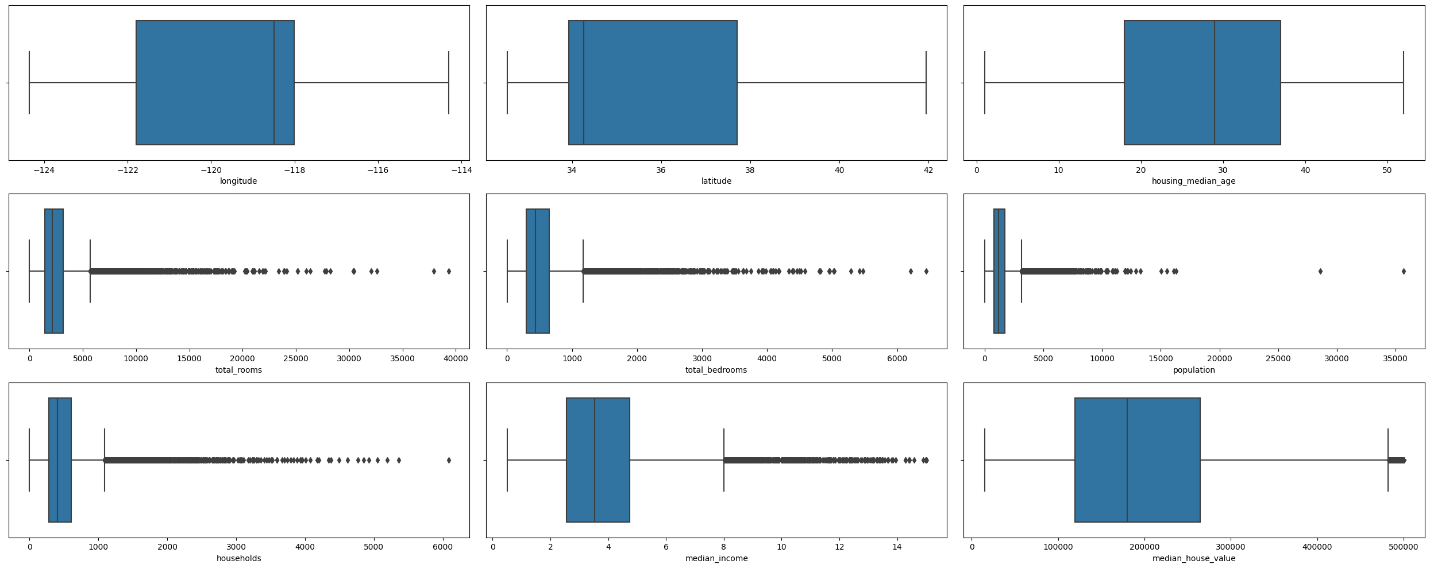
توزیع نسبتاً یکنواخت به نظر می رسد، با تمرکز جزئی در سنین میانی. این نشان دهنده تنوع سن خانه‌هاست.

Total Rooms, Total Bedrooms, Population, و Households:

این نمودارها توزیعی به سمت راست را نشان می‌دهند که از سبیل‌های بلندی که به سمت راست امتداد می‌یابند و خط میانی به سمت چپ جعبه مشهود است.



شکل 17: کد رسم کننده نمودارهای جعبه‌ای



شکل 18: نمودارهای جعبه‌ای

سبیل‌های بلند محدوده ای از مقادیر بزرگتر را نشان می دهد، در حالی که کادر نشان می دهد که قسمت عمده داده ها کجا قرار دارد. این نشان می‌دهد که در حالی که اکثر املاک دارای محدوده کمتری از اتاق‌ها، اتاق‌خواب‌ها، جمعیت و اندازه‌های خانوار هستند، اما نقاط پرت قابل توجهی با مقادیر بسیار بالا وجود دارد.

این الگو اغلب نشان می‌دهد که چند ملک بسیار بزرگ یا مناطق پرجمعیت وجود دارد، اما اکثریت خانه‌ها، اتاق‌های کمتر و جمعیت/خانوارهای کوچک‌تری دارند.

Median Income:

درآمد متوسط دارای چولگی به راست است اما چولگی کمتری نسبت به کل اتاق ها یا اتاق خواب ها دارد. مقادیر پرت کمتری وجود دارد که نشان دهنده توزیع یکنواخت درآمد در میان مجموعه داده است.

Median House Value:

توزیع ارزش خانه دارای دامنه وسیعی است، با این جعبه که بسیار گسترده است، که نشان می دهد تغییرات قابل توجهی در قیمت خانه وجود دارد. میانه به چارک پایین نزدیکتر است، که نشان می دهد خانه های بیشتری کمتر از میانه قیمت گذاری شده اند.

سبیل بلند سمت راست نشان می دهد که خانه هایی با ارزش به طور قابل توجهی بالاتر از میانه وجود دارد که نشان دهنده وجود املاک لوکس یا با ارزش بالا است.

4-2-1-رسم و تفسیر ماتریس همبستگی

برای رسم ماتریس، ابتدا باید داده‌های دسته‌ای را one hot encode کرد. برای این منظور از متد get\_dummies() استفاده می‌کنیم. سپس ستون هدف و ستون ocean\_proximity را حذف می‌کنیم و ابتدا ستون‌های انکود شده و سپس ستون هدف را به دیتافریم می‌چسبانیم. برای این منظور از تابع pd.concat() استفاده می‌کنیم. در ادامه با استفاده از متد corr() ماتریس همبستگی را ایجاد می‌کنیم. شکل (19) نحوه انجام این کار را نمایش می‌دهد.



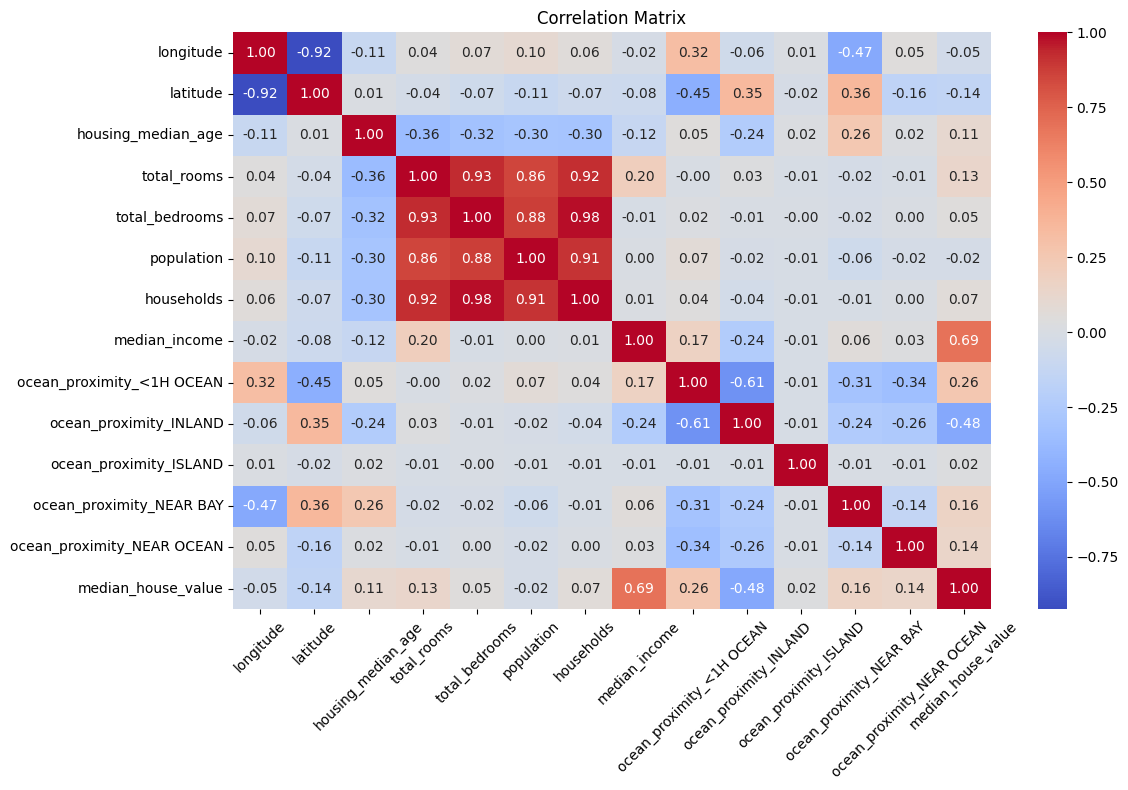
شکل 19: آماده‌سازی دیتافریم برای محاسبه ماتریس همبستگی

سپس با استفاده از کتابخانه seaborn و تابع heatmap() ماتریس ایجاد شده را رسم می‌کنیم. شکل (20) نحوه رسم این ماتریس را نمایش می‌دهد.



شکل 20: رسم ماتریس همبستگی

همچنین شکل (21) ماتریس رسم شده را نمایش می‌دهد.



شکل 21: ماتریس همبستگی رسم شده

**تفسیر:**

طول و عرض جغرافیایی: یک همبستگی منفی قوی بین طول و عرض جغرافیایی وجود دارد، که نشان می دهد در این مجموعه داده جغرافیایی، با رفتن به سمت غرب (با افزایش طول جغرافیایی)، فرد تمایل به رفتن به سمت جنوب (کاهش عرض جغرافیایی) دارد یا برعکس.

میانه سن مسکن: به نظر می‌رسد که با اکثر متغیرهای دیگر همبستگی کمی یا بدون وجود دارد، که نشان می‌دهد سن خانه‌ها رابطه خطی قوی با ویژگی‌هایی مانند اتاق‌ها، اتاق‌خواب‌ها یا اندازه خانه ندارد.

مجموع اتاق‌ها/اتاق‌های خواب و خانوارها/جمعیت: همبستگی‌های مثبت قوی در این گروه‌ها وجود دارد، که شهودی است زیرا اتاق‌های بیشتر معمولاً با تعداد اتاق‌خواب‌های بیشتر مرتبط هستند و اندازه جمعیت بزرگ‌تر با تعداد خانوارهای بیشتر مرتبط است.

درآمد متوسط و میانه ارزش خانه: یک همبستگی مثبت قابل توجه وجود دارد، که نشان می دهد مناطق درآمد متوسط بالاتر با ارزش خانه بالاتر مرتبط هستند، که یک الگوی اقتصادی رایج است.

نزدیکی اقیانوس: دسته‌های مختلف نزدیکی اقیانوس درجات مختلفی از همبستگی با میانگین ارزش خانه دارند. این نشان می دهد که نزدیکی به اقیانوس ممکن است قیمت مسکن را به طور متفاوت تحت تاثیر قرار دهد، به طوری که برخی از مقوله ها دارای همبستگی مثبت و برخی دیگر منفی هستند.

### 5-2-1-رسم و تفسیر نمودار همبستگی با ستون هدف

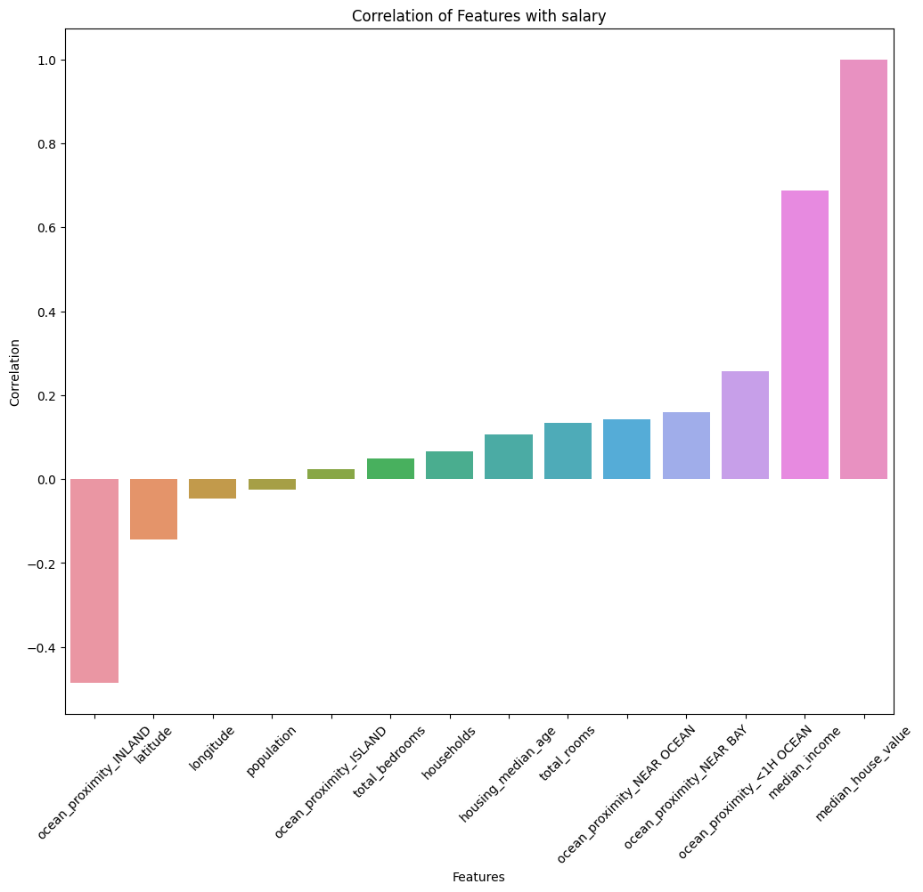
برای رسم این نمودار، ابتدا با استفاده از متد corrwith() همبستگی ستون‌ها با ستون هدف را محاسبه می‌کنیم. سپس این مقادیر را در متغیر correlation ذخیره می‌کنیم. سپس مقادیر موجود در این لیست را با متد sort\_values() به طور صعودی مرتب می‌کنیم. و با استفاده از تابع barplot() از کتابخانه seaborn از این مقادیر نمودار میله‌ای رسم می‌کنیم. شکل (22) کدی که وظیفه‌ی انجام این عمل را به عهده دارد را نمایش می‌دهد. همچنین شکل (23) خروجی این کد را نمایش می‌دهد.

تفسیر:

همبستگی منفی (نوارهای قرمز و نارنجی):



شکل 22: کد رسم کننده نمودار میزان همبستگی با سطر هدف



شکل 23: نمودار میزان همبستگی با سطر هدف

ویژگی‌های «ocean\_proximity\_INLAND» و «Latitude» همبستگی منفی با ستون هدف دارند، که نشان می‌دهد با افزایش این مقادیر ویژگی، حقوق تمایل به کاهش دارد.

'ocean\_proximity\_INLAND' قوی ترین همبستگی منفی را در بین ویژگی های نمایش داده شده دارد.

همبستگی ضعیف تا بدون (نوارهای زرد تا آبی روشن):

ویژگی‌هایی مانند «longitude»، «population»، «ocean\_proximity\_ISLAND»، «households»، «total\_bedrooms»، « housing\_median\_age» و «total\_rooms» همبستگی ضعیف و بدون هیچ ارتباطی با ستون هدف نشان می‌دهند.

همبستگی مثبت (نوارهای آبی تا صورتی):

ویژگی‌های «ocean\_proximity\_NEAR BAY» و «ocean\_proximity\_<1H OCEAN» همبستگی مثبتی دارند، که نشان می‌دهد با افزایش نزدیکی به این مناطق، میزان ستون هدف نیز افزایش می‌یابد.

« median\_income» و « median\_house\_value» همبستگی مثبت قوی با ستون هدف نشان می‌دهند.

قوی ترین همبستگی ها (نوارهای صورتی):

میله 'median\_income' بسیار بلند است، که یک همبستگی مثبت قوی را نشان می دهد، که به طور بالقوه نشان می دهد که این ویژگی می تواند پیش بینی خوبی برای ستون هدف باشد.

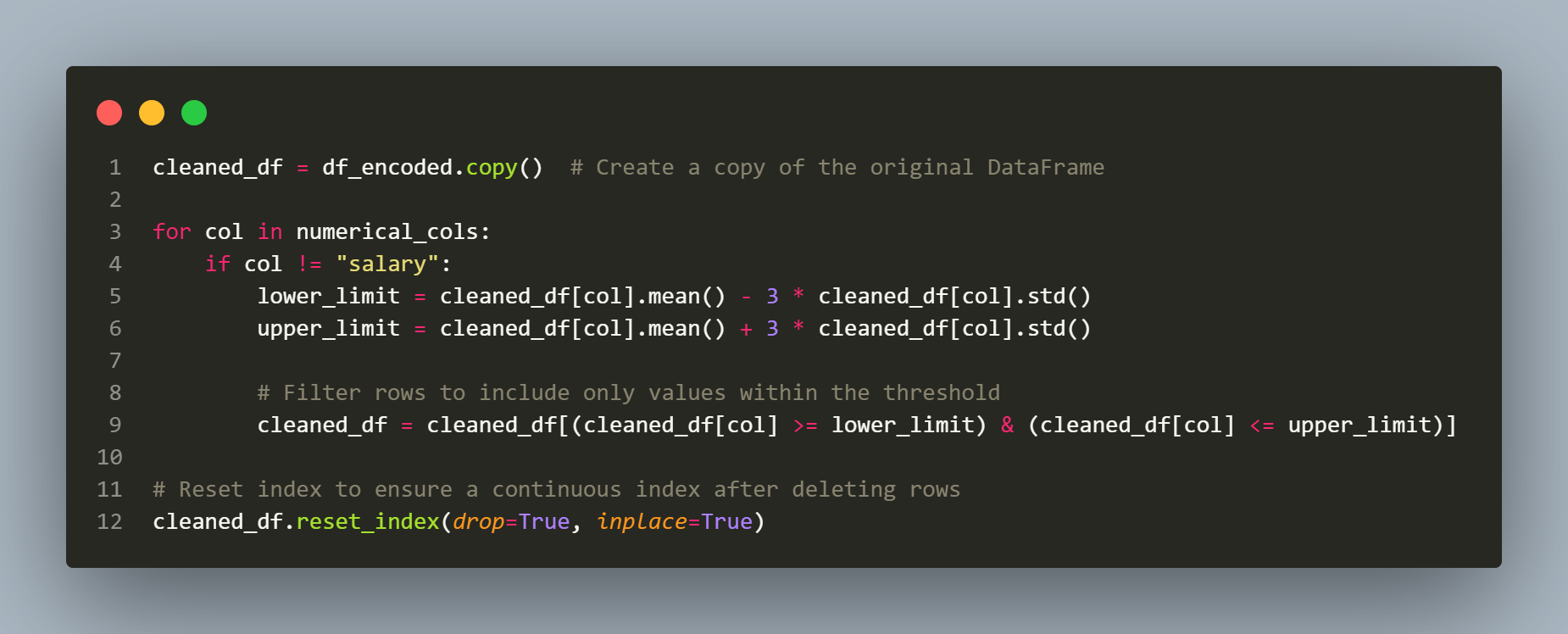
## 3-1-تشخیص داده‌های پرت

برای تشخیص داده‌های پرت از روش شش سیگما استفاده می‌کنیم. نحوه کارکرد این روش به این صورت است که برای هر ستون میانگین و واریانس محاسبه می‌کنیم و برای هر ستون، هر داده‌ای که کمتر از میانگین منهای سه سیگما و بزرگتر از میانگین به علاوه سه سیگما بود را به عنوان داده پرت شناسایی می‌کنیم. برای مدیریت داده‌های پرت دو رویکرد را با هم مقایسه می‌کنیم. این دو رویکرد عبارت است از حذف داده‌های پرت یا جایگزینی آنها با حد پایین یا حد بالا.

دیتاست حاصل از این دو روش را با الگوریتم رگرسور جنگل تصادفی و ارزیابی مقدار R2 مقایسه می‌کنیم.

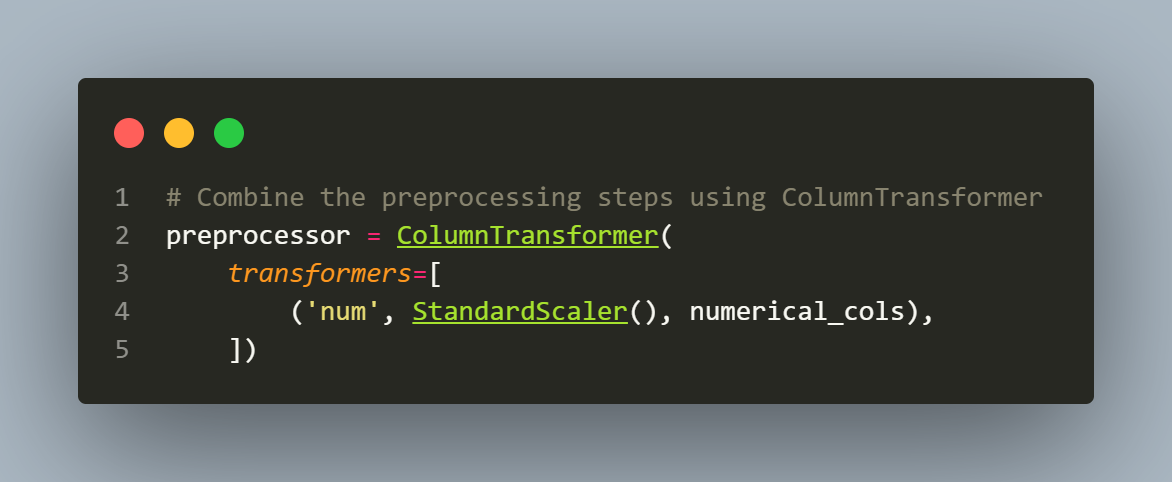
### 1-3-1-جایگزینی داده‌ها با حد پایین یا حد بالا

ابتدا از دیتافریم به دست آمده از بخش قبلی، یک کپی در متغیر cleaned\_df ذخیره می‌کنیم، حالا با استفاده از یک حلقه، برای هر ستون میانگین، واریانس، حد پایین و حد بالا را محاسبه می‌کنیم. سپس مقادیری که بیشتر از حد بالا هستند را با مقدار حد بالا و مقادیری که کمتر از حد پایین هستند را با حد پایین جایگزین می‌کنیم. شکل (24) نحوه انجام این کار را نمایش می‌دهد.



شکل 24: جایگزینی داده‌های پرت با حد بالا و حد پایین

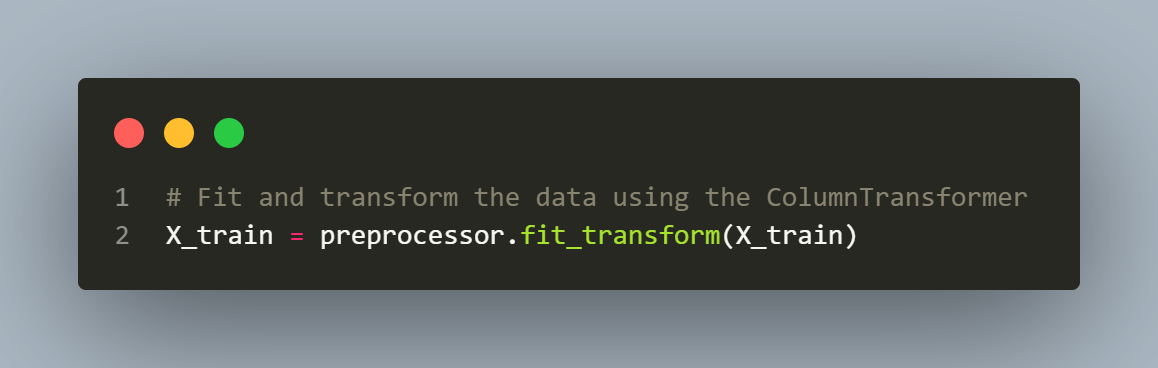
پس از تقسیم داده‌ها به دسته‌های آموزش و تست، داده‌ها را استاندارد سازی می‌کنیم. برای این منظور یک شی از کلاس ColumnTransformer() به نام preprocessor تعریف می‌کنیم. این شی می‌تواند یک یا چند عمل را ستون به ستون روی دیتافریم انجام دهد. سپس یک تاپل که شامل نام عملیات، کلاسی که عملیات را انجام می‌دهد و ستون‌هایی که این عملیات‌ها باید روی آنها انجام شود را به آن پاس می‌دهیم. این کار باعث می‌شود که در صورت اجرای متد fit() از این شی، عملیات تعریف شده، ستون به ستون روی دیتافریم اجرا شوند. شکل (25) این کلاس و شی ایجاد شده از آن را نمایش می‌دهد.



شکل 25: ایجاد شی از کلاس ColumnTransformer

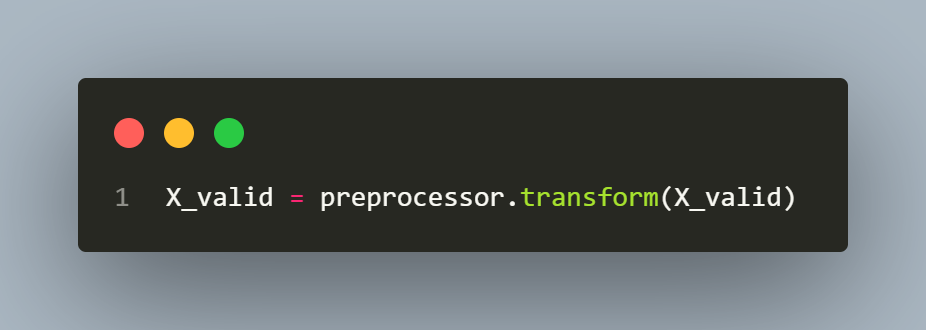
در ادامه از شی preprocessor متد .fit\_transform() را فراخوانی می‌کنیم و دیتاست آموزشی را به آن پاس می‌دهیم. این متد پارامتر دیتاستی که به عنوان آرگومان دریافت کرده را یاد می‌گیرد، سپس با توجه به پارامترهایی که محاسبه کرده، عملیات‌های تعریف شده را انجام می‌دهد. و سپس نتیجه را در همان متغیر قبلی ذخیره می‌کنیم. شکل (26) نحوه عملکرد این کد را نمایش می‌دهد.

سپس برای پیش‌پردازش داده‌های تست، از متد transform() شی استفاده می‌کنیم. دلیل استفاده از این متد این است که این متد با استفاده از پارامترهای یادگرفته شده از داده‌های آموزشی، عملیات را روی داده‌های تستی انجام می‌دهد. توجه شود که اینجا از متد قبلی استفاده نمی‌شود



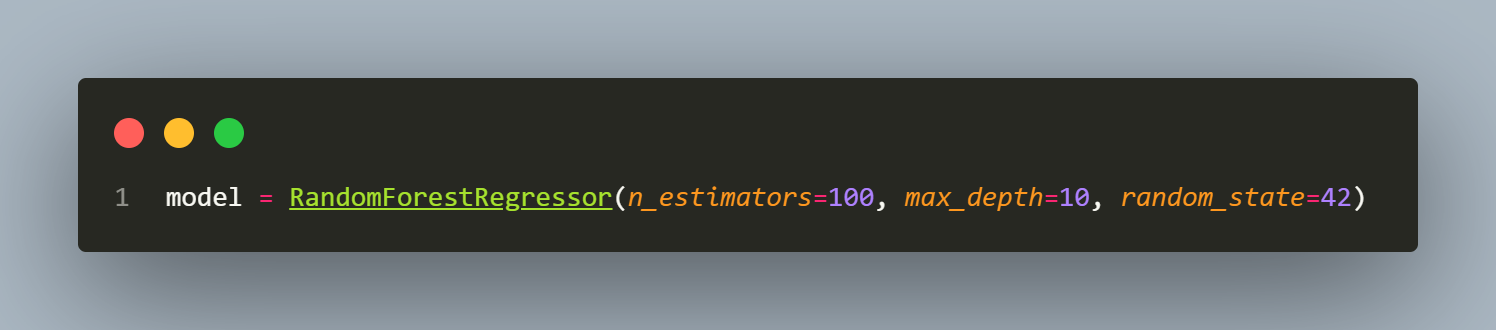
شکل 26: پیش‌پردازش داده‌های آموزشی

زیرا فرض بر این است که پارامترهای داده‌های تست را نداریم و باید با استفاده از برآوردی که در داده‌های آموزشی انجام دادیم، عملیات‌ها را روی داده‌های تست انجام دهیم. عدم رعایت این مورد، موجب ایجاد نشت داده می‌شود. شکل (27) نحوه پیش‌پردازش داده‌های تست را نمایش می‌دهد.



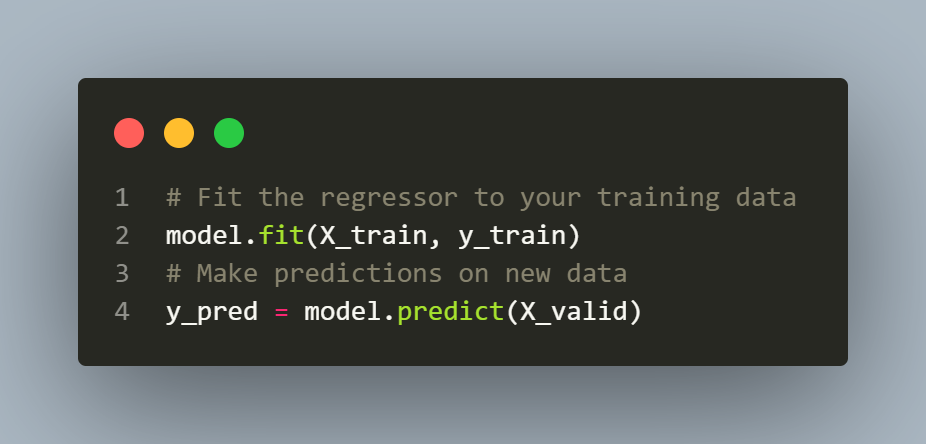
شکل 27: پیش‌پردازش داده‌های تست

سپس مدل را تعریف می‌کنیم. یک شی از کلاس RandomForestRegressor() ایجاد می‌کنیم و آن را model می‌نامیم. حداکثر عمق درخت‌ها را 10، و تعداد برآورد کننده ها را برابر 100 قرار می‌دهیم. همچنین برای امکانپذیری بازتولید نتایج، حالت رندوم را برابر 42 قرار می‌دهیم. شکل (28) تعریف مدل را نمایش می‌دهد.



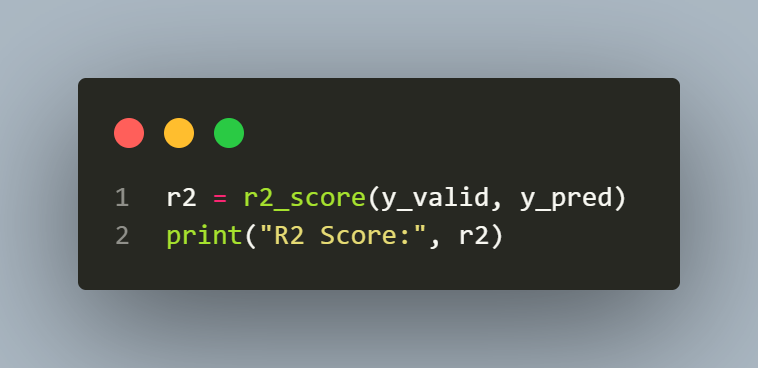
شکل 28: تعریف مدل

سپس مدل را با استفاده از متد .fit() روی داده‌های آموزشی آموزش می‌دهیم و با پاس دادن داده‌های تست به متد .predict() ، پیش‌بینی مدل را در متغیر y\_pred ذخیره می‌کنیم. شکل (29) نحوه انجام این دو عملیات را نمایش می‌دهد.



شکل 29: آموزش و تست مدل

سپس مقدار R2 را محاسبه می‌کنیم. شکل (30) نحوه انجام این کار با تابع r2\_score() را نمایش می‌دهد.

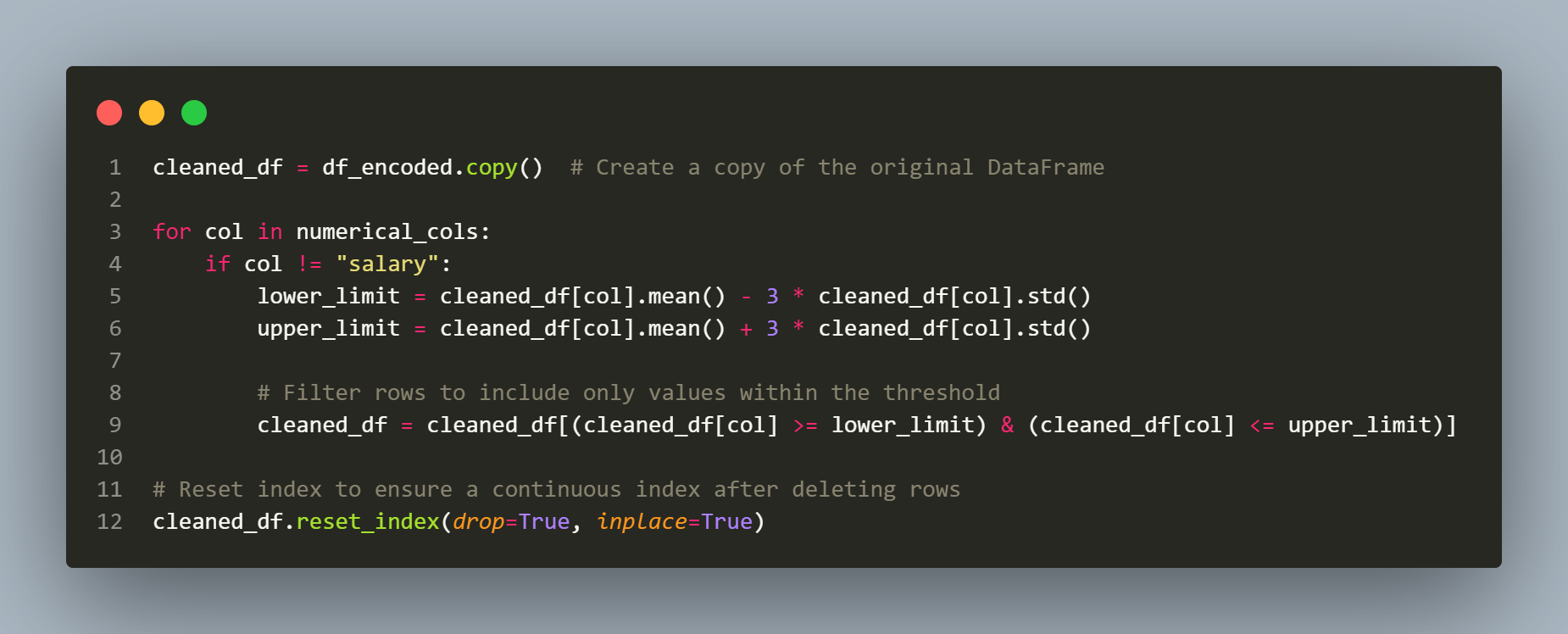


شکل 30: محاسبه و نمایش R2

با روش جایگزینی مقادیر پرت، مقدار R2 برابر 0.7770 می‌شود. حال روش حذف داده‌های پرت را ارزیابی می‌کنیم.

2-3-1-حذف داده‌های پرت

همانند بخش قبلی روی یک کپی از دیتاست بخش قبلی کار خود را انجام می‌دهیم. سپس داده‌]ایی که به به عنوان پرت شناخته می‌شوند را حذف می‌کنیم. شکل (31) نحوه انجام این کار را نمایش می‌دهد.



شکل 31: حذف داده‌های پرت

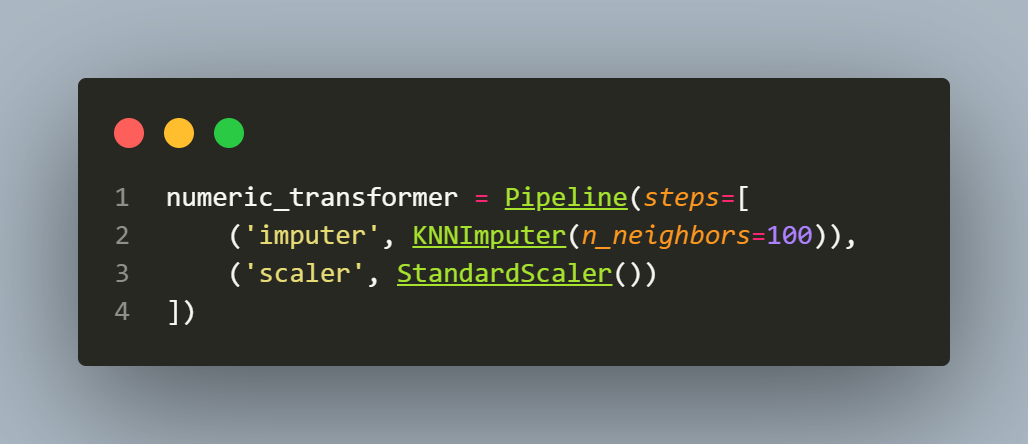
سپس تمامی مراحل را مشابه بخش قبلی انجام می‌دهیم و میزان R2 را محاسبه می‌کنیم. در این روش، میزان R2 برابر 0.7501 خواهد شد. بنابراین برای مدیریت داده‌های پرت، از روش حذف آنها استفاده می‌کنیم.

## 4-1-پر کردن مقادیر خالی

برای پر کردن مقادیر خالی دو روش را با هم مقایسه می‌کنیم. اولین روش پر کردن مقادیر با استفاده از KNN است و روش دوم استفاده از میانگین است. دیتاست حاصل از این دو روش را با الگوریتم رگرسور جنگل تصادفی و ارزیابی مقدار R2 مقایسه می‌کنیم.

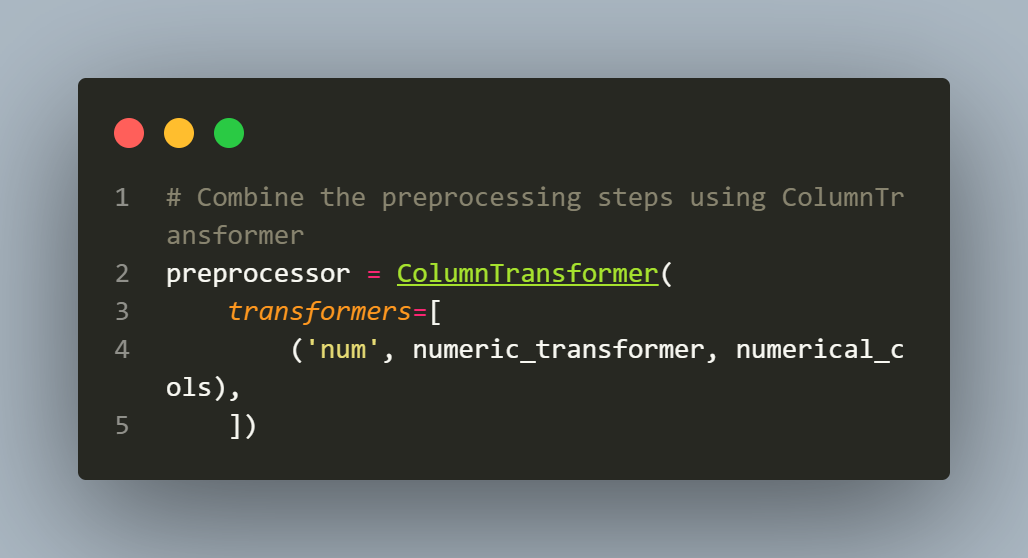
### 2-4-1-تعریف شی پیش‌پردازش برای پر کردن مقادیر خالی با KNN

ابتدا داده‌های تمیز شده که در دیتافریم cleaned\_df قرار گرفته‌اند را به دو دسته آموزش و تست تقسیم می‌کنیم. در ادامه برای انجام راحت پیش‌پردازش یک شی از کلاس پایپ‌لاین تعریف می‌کنیم. این شی دسته‌ای از عملیات‌ها را به ترتیب روی هدف انجام می‌دهد. شی تعریف شده در این قسمت، ابتدا مقادیر خالی را با استفاده از تابع KNNImputer() پر می‌کند و سپس داده‌ها را استاندارد سازی می‌کند. شکل (32) کد مربوط به این شی را نمایش می‌دهد.



شکل 32: ایجاد شی از کلاس پایپ‌لاین

سپس همانند بخش قبلی یک شی از کلاس ColumnTransformer() ایجاد می‌کنیم و در عضو سوم، شی ساخته شده از پایپ‌لاین را قرار می‌دهیم. شکل (33) ترنسفورمر ایجاد شده را نمایش می‌دهد.

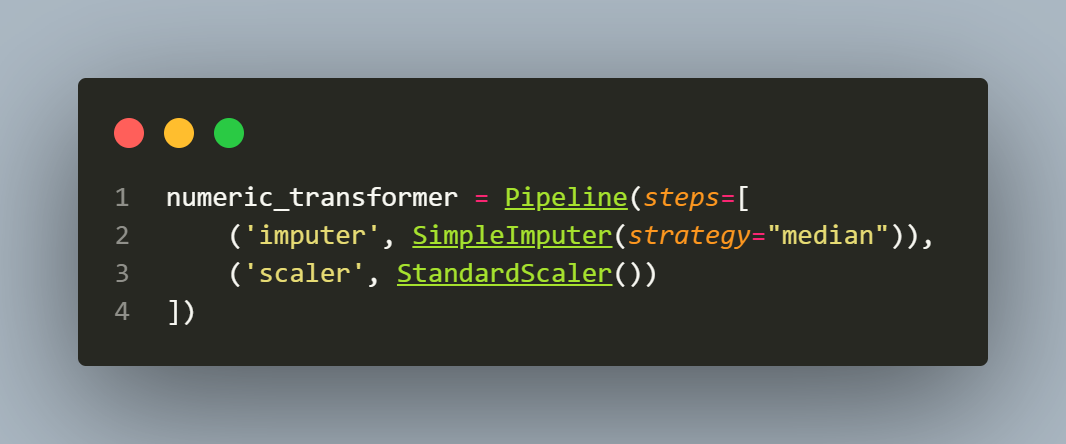


شکل 33: ایجاد شی از کلاس ColumnTransformer

سپس همانند بخش قبلی داده‌های آموزش و تست را پیش‌پردازش می‌کنیم، و مدل را آموزش می‌دهیم. سپس پیش‌بینی را انجام می‌دهیم و مقدار R2 را محاسبه می‌کنیم. در این روش، R2 برابر با 0.7694 می‌شود.

### 4-4-1-تعریف شی پیش‌پردازش برای پر کردن مقادیر خالی با میانه

حال شی تعریف شده برای پیش‌پردازش را به نحوی تغییر می‌دهیم که مقادیر خالی را با میانه پر کند. برای این منظور، در شی پایپ‌لاین، به جای KNNImputer() کلاس SimpleImputer() با آرگومان strategy = ‘median’ را پاس می‌دهیم. شکل (34) پایپ‌لاین به روز شده را نمایش می‌دهد.



شکل 34: پایپ‌لاین به روز شده

سایر مراحل را مشابه بخش قبلی انجام می‌دهیم. و مدل را آموزش و تمرین می‌دهیم و مقدار R2 را محاسبه می‌کنیم. در این حالت مقدار R2 برابر 0.7694 خواهد شد. هر دو روش دقیقاً یک مقدار R2 را می‌دهند. بنابراین تفاوتی بین انتخاب این دو روش وجود ندارد. از آنجایی که روش پر کردن داده‌ها با میانه به مراتب راحت‌تر است، مقادیر خالی را با میانه پر می‌کنیم.

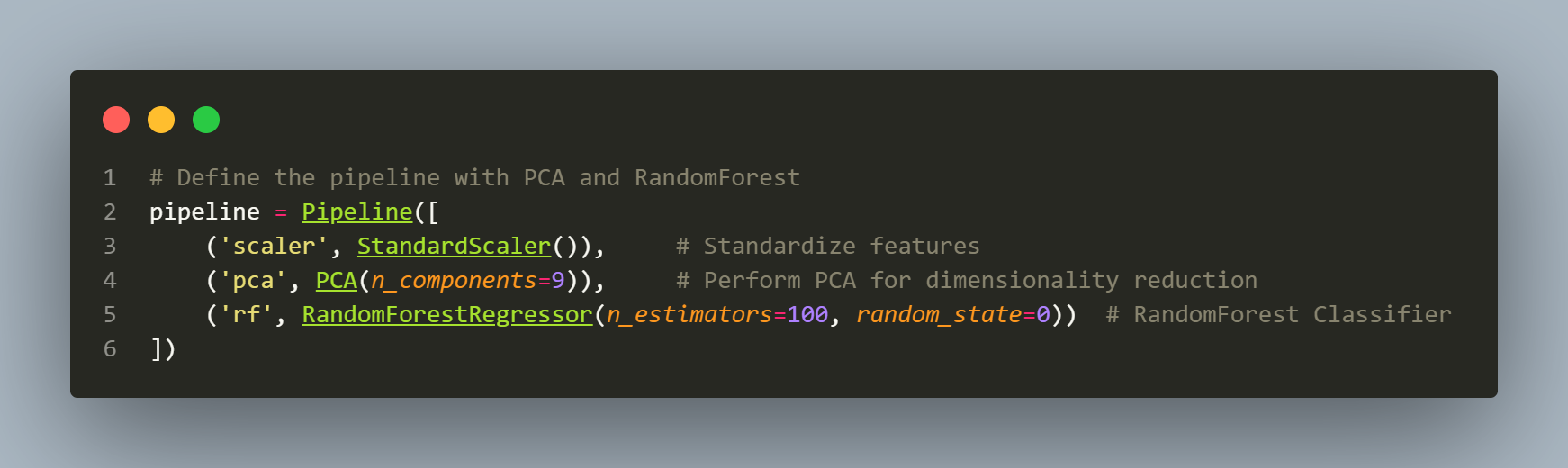
در ادامه به انتخاب ویژگی می‌پردازیم.

## 5-1-انتخاب ویژگی

برای انتخاب ویژگی از دو روش استفاده می‌کنیم. روش اول استفاده از PCA رو روش دوم استفاده از Mutual Information است. سپس این دو روش را همانند روشی که در دو قسمت قبلی استفاده شد، با هم مقایسه می‌کنیم.

### 1-5-1- انتخاب ویژگی با استفاده از روش PCA

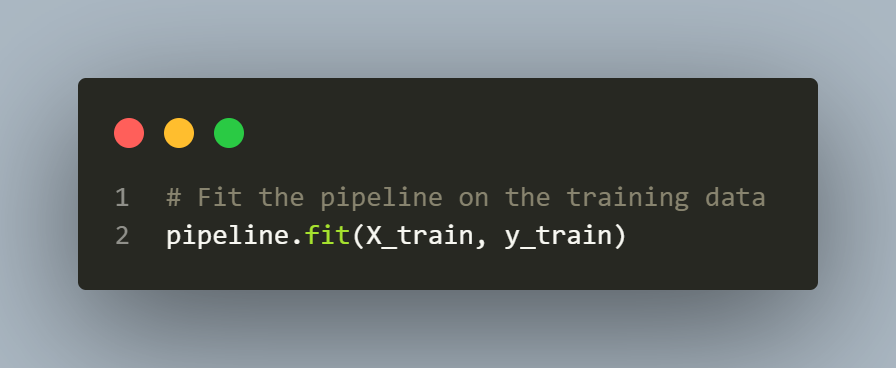
ابتدا دیتاست به دست آمده از بخش قبلی را به دو بخش آموزش و تست تقسیم می‌کنیم. سپس یک پایپ‌لاین ایجاد می‌کنیم که در آن به ترتیب داده‌ها استانداردسازی می‌شوند، سپس 9 مولفه اساسی انتخاب می‌شوند و روی مولفه‌های انتخاب شده، یک الگوریتم رندوم فارست با پارامترهای بخش ‌های قبلی آموزش داده می‌شود. شکل (35) این پایپ‌لاین را نمایش می‌دهد.



شکل 35: پایپ‌لاین ایجاد شده برای PCA

سپس با استفاده از متد fit() عملیات‌های گفته شده را روی دیتاست آموزشی اجرا می‌کنیم شکل (36) استفاده از این متد را نمایش می‌دهد.

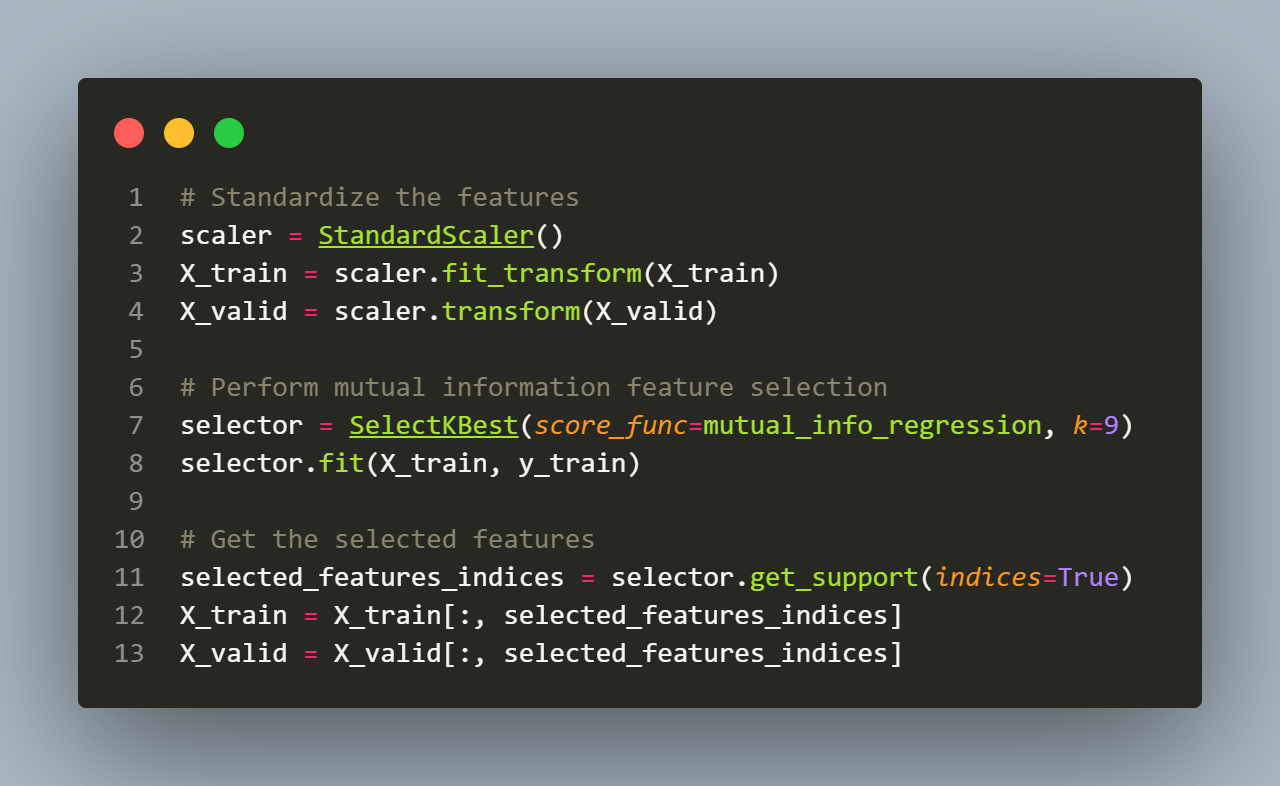
سپس با استفاده از متد .predict() و پاس دادن داده‌های تست، عملیات‌های گفته شده را روی داده‌های تست انجام می‌دهیم و نتایج پیش‌بینی را ذخیره می‌کنیم. سپس با مقایسه پیش‌بینی و مقادیر حقیقی، مقدار R2 را محاسبه می‌کنیم. در این روش مقدار R2 برابر 0.7694 می‌شود



شکل 36: اجرای پایپ‌لاین روی داده‌های آموزشی

### 2-5-1- انتخاب ویژگی با روش Mutual Information

ابتدا داده‌های آموزشی و تستی را استاندارد سازی می‌کنیم. سپس با استفاده از کلاس SelectKBest() و آرگومان‌های score\_func=mutual\_info\_regression و k=9 یک شی می‌سازیم که متد .fit() آن با استفاده از Mutual Information نه ویژگی برتر دیتاست را انتخاب می‌کند. سپس با متد get\_support() اندیس این ویژگی‌ها را در لیستی به نام selected\_features\_indices ذخیره می‌کنیم و فقط ستون‌های حاضر در این لیست را نگه می‌داریم. شکل (37) نحوه انجام این کار را نمایش می‌دهد.



شکل 37: انتخاب نه ویژگی برتر با استفاده از Mutual Information

سپس همانند بخش‌های قبلی یک مدل جنگل تصادفی روی داده‌ها آموزش می‌دهیم و تست می‌کنیم. در این روش R2 به دست آمده برابر 0.752 است. در نتیجه برای انتخاب ویژگی از روش Mutual Information استفاده می‌کنیم.

حال دیتاستی که تا این مرحله پیش‌پردازش شده است را مجدداً در متغیر df ذخیره می‌کنیم.

6-1- پیدا کردن درجه مناسب رگرسیون

در این بخش به بررسی این موضوع می‌پردازیم که کدام درجه از رگرسیون نتیجه‌ی بهتری حاصل می‌کند. به این منظور از Cross Validation استفاده می‌کنیم.

ابتدا دیتافریم را به دو دسته آموزشی و تستی تقسیم می‌کنیم. سپس لیستی از درجات درست می‌کنیم. درجات از یک تا پنج خواهند بود. از آنجایی که زمان آموزش مدل به طور نمایی بالا می‌رود، آموزش مدل از درجه شش و بالاتر نیازمند صرف زمان بسیار زیاد است و نتیجه به دست آمده بهبود نخواد داشت.

برای ارزیابی از روش MSE استفاده می‌کنیم. برای اینکه امکان رسم نمودار MSE بر اساس درجات را داشته باشیم، یک لیست خالی به نام mse\_values\_list ایجاد می‌کنیم. سپس یک حلقه ایجاد می‌کنیم که به ازای هر یک از درجات به ترتیب این موارد را انجام می‌دهد:

1-ابتدا یک پایپ‌لاین ایجاد می‌کند که این پایپ‌لاین به ترتیب داده‌ها را استاندارد می‌کند، سپس درجه‌ مربوطه که توسط حلقه تعریف می‌شود را می‌سازد و مدل رگرسیون را آموزش می‌دهد.

2-یک لیست خالی تعریف می‌شود که مسئول نگهداری مقدار MSE به دست آمده از هر بخش است.

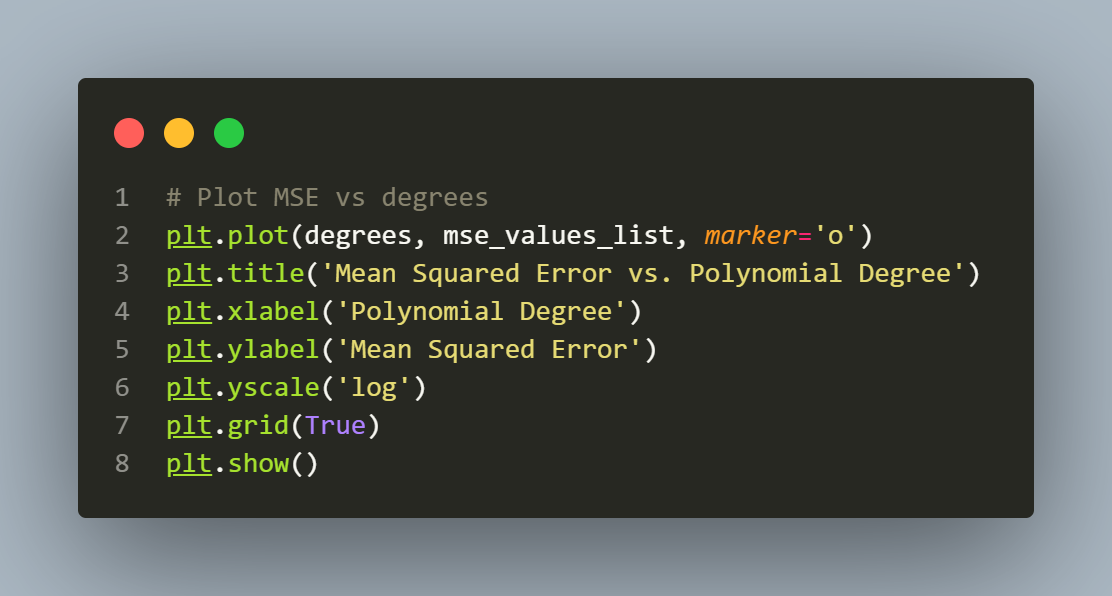
3-از کلاس KFold یک شی تعریف می‌شود. این شی دیتاست آموزشی را بر می‌زند و به پنج بخش تقسیم می‌کند. سپس برای هر بخش، دیتای آموزشی با استفاده از پایپ‌لاین تعریف شده پیش‌پردازش می‌شود و سپس مدل رگرسیون روی آن آموزش می‌بیند. در ادامه مجدداً با استفاده از پایپ‌لاین، داده‌های تست پیش‌پردازش می‌شوند و پیش‌بینی انجام می‌شود. و در نهایت MSE محاسبه می‌شود و در لیست ایجاد شده ذخیره می‌شود.

4- این عمل روی تمامی بخش ها انجام شده و حلقه به پایان می‌رسد. MSE میانگین این بخش‌ها محاسبه شده و در لیست mse\_values\_list قرار می‌گیرد.

5- تمامی عملیات های 1 تا 4 برای درجه بعدی تکرار می‌شود.

شکل (39) نحوه انجام این عملیات را نمایش می‌دهد.

سپس از مقادیر به دست آمده نمودار رسم می‌کنیم. برای نمایش بهتر مقادیر MSE، محور y را به صورت لگاریتمی رسم می‌کنیم. شکل (38) کدی که وظیفه رسم این نمودار را دارد را نمایش می‌دهد.

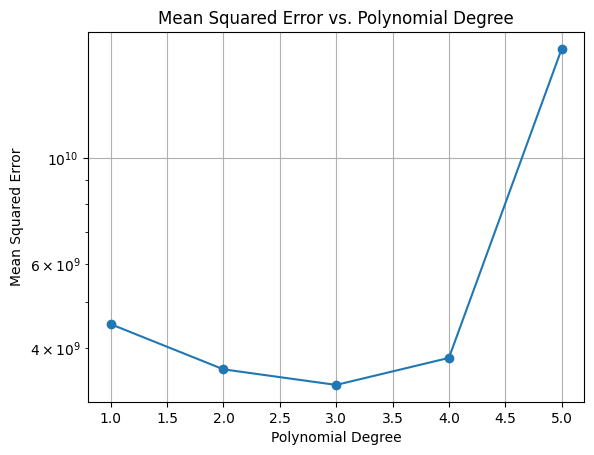


شکل 38: رسم نمودار از میانگین MSE هر درجه



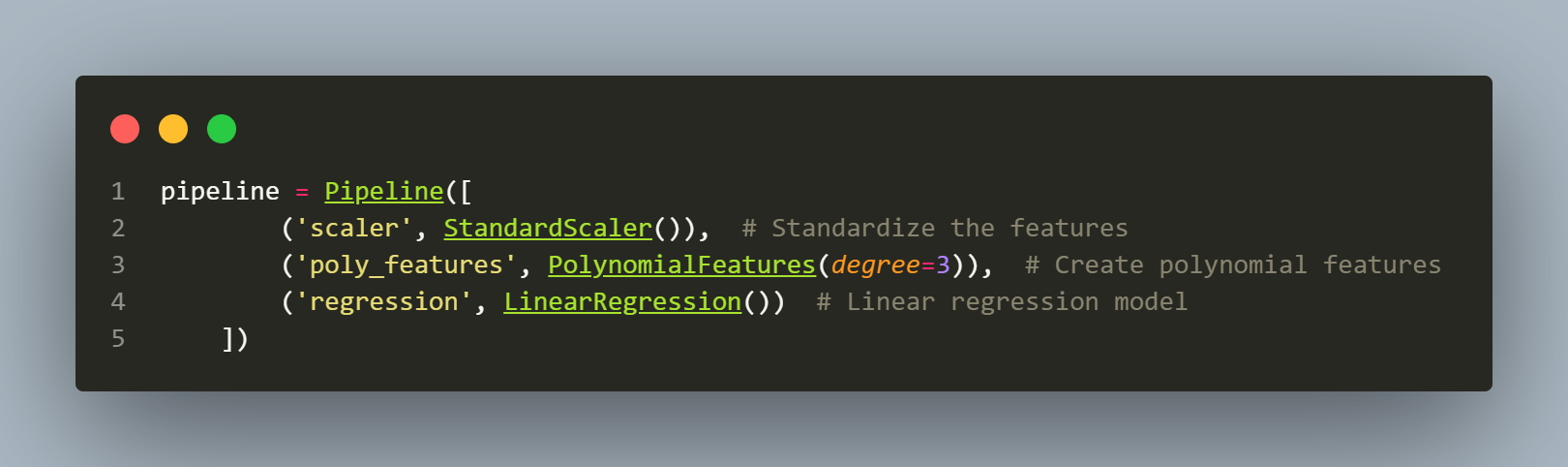
شکل 39: ارزیابی درجه‌های رگرسیون با استفاده از CV

همچنین شکل (40) نمودار رسم شده را نمایش می‌دهد.



شکل 40: نمودار MSE به ازای درجه

با توجه به نمودار، مشخص است که درجه سه کمترین MSE را دارد. در نتیجه مدل را با درجه سه آموزش می‌دهیم. برای آموزش و تست مدل از همان پایپ‌لاین ایجاد شده در حلقه استفاده می‌کنیم. شکل (41) این پایپ‌لاین را نمایش می‌دهد.

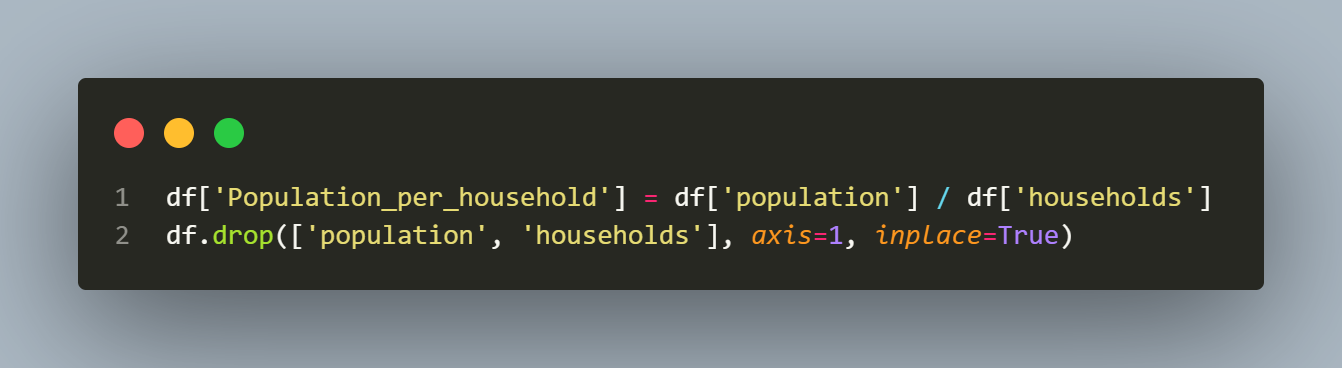


شکل 41: پایپ‌لاین پیش‌پردازش و آموزش رگرسیون

سپس همانند بخش‌های قبلی MSE و R2 را محاسبه می‌کنیم. این مقادیر به ترتیب 3228540927.528902 و 0.7397 به دست می‌آیند.

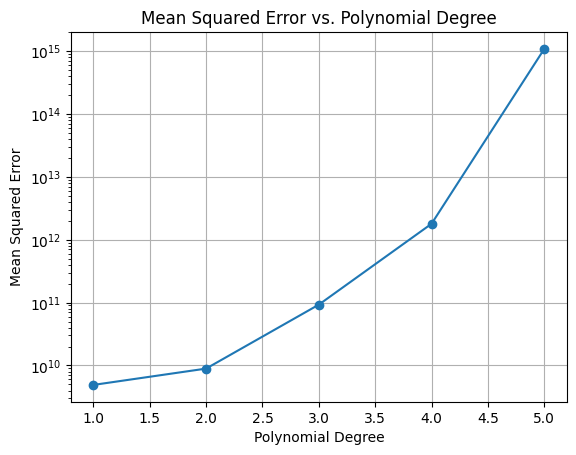
7-1-امتحان مجدد مدل، با استفاده از دو ویژگی جدید

ابتدا ویژگی Population\_per\_household را بررسی می‌کنیم. برای ایجاد این ویژگی دو ستون متناظر را بر هم تقسیم می‌کنیم و در ستونی با اسم همین ویژگی قرار می‌دهیم و این دو ویژگی را حذف می‌کنیم. شکل (42) نحوه انجام این کار را نمایش می‌دهد.



شکل 42: ایجاد ویژگی جدید با دو ویژگی قدیمی

سپس همه مراحل را مانند قسمت قبل تکرار می‌کنیم. نمودار MSE بر اساس درجه برای این حالت، مانند شکل (43) خواهد بود.

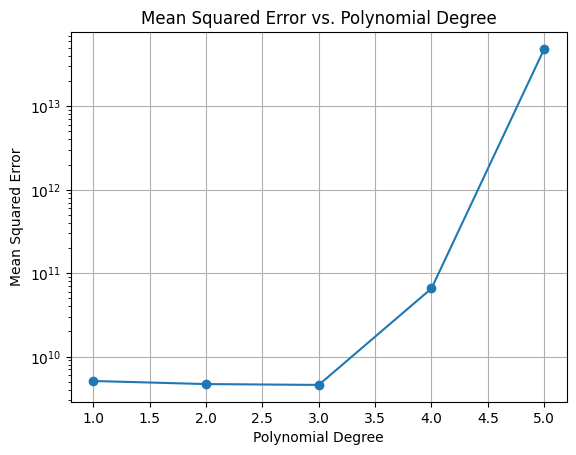


شکل 43: نمودار MSE بر درجه برای حالت جدید دیتاست

سپس با همان روش قبلی یک رگرسیون خطی آموزش و تست و ارزیابی می‌کنیم. در این حالت مقدار MSE و R2 به ترتیب برابر 4831621435.534403 و 0.6104 خواهد شد.

مشاهده می‌شود که MSE افزایش داشته ولی R2 کاهش یافته است.

سپس ویژگی Rooms\_per\_household را بررسی می‌کنیم. نحوه ایجاد این ویژگی کاملاً مانند ویژگی قبلی است. در دیتاست اصلی این ویژگی را ایجاد می‌کنیم و سپس نمودار MSE بر اساس درجات را ایجاد می‌کنیم که شکل (44) آن را نمایش می‌دهد.



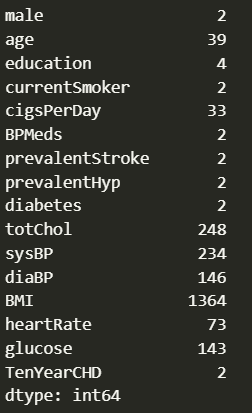
شکل 44: نمودار MSE براساس درجه در حالت داشتن ویژگی Rooms\_per\_household

با توجه به شکل مجدداً درجه سوم بهترین عملکرد را دارد ولی عملکرد درجه دو و درجه یک هم بهبود قابل توجهی داشته در حدی که اختلاف این دو درجه با درجه سه بسیار کم است.

سپس با همان روش قبلی یک رگرسیون خطی آموزش و تست و ارزیابی می‌کنیم. در این حالت مقدار MSE و R2 به ترتیب برابر 4419074284.970041 و 0.643 خواهد بود.

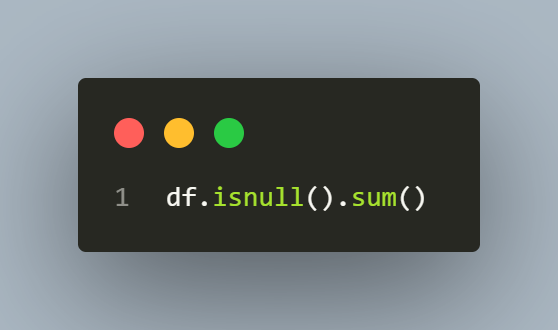
# 2-پاسخ سوال دوم

ابتدا دیتاست را لود می‌کنیم. سپس با استفاده از متد .info() دیتاست را بررسی می‌کنیم. برخی از ستون‌های دیتاست، داده‌های گمشده دارند. همچنین همه ستون‌ها دارای دیتاتایپ int یا float هستند حال از متد .nunique() برای بررسی تعداد مقادیر یکتا در هر ستون استفاده می‌کنیم. مشاهده می‌شود که برخی از ستون‌ها فقط دو مقدار یکتا دارند که نشان‌دهنده‌ی دسته‌ای بودن این ستون‌ها است. همچنین ستون education فقط چهار مقدار یکتا دارد که نشان می‌دهند یک ستون دسته‌ای با کاردینالیتی چهار است. شکل (45) خروجی متد .nunique() را نمایش می‌دهد.



شکل 45: خروجی متد .nunique()

حال به بررسی دقیق تعداد مقادیر خالی در هر دسته می‌پردازیم. برای این منظور از method chaining استفاده می‌کنیم. به این صورت که روی متغیر df دو متد .isnull() و .sum() را اجرا می‌کنیم. شکل (46) این method chaining را نمایش می‌دهد.



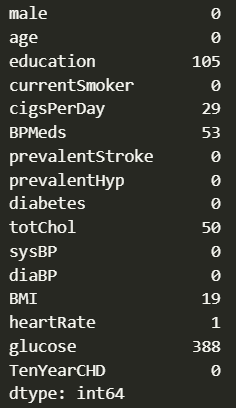
شکل 46: استفاده از method chaining

خروجی این عملیات در شکل (47) نمایش داده شده.

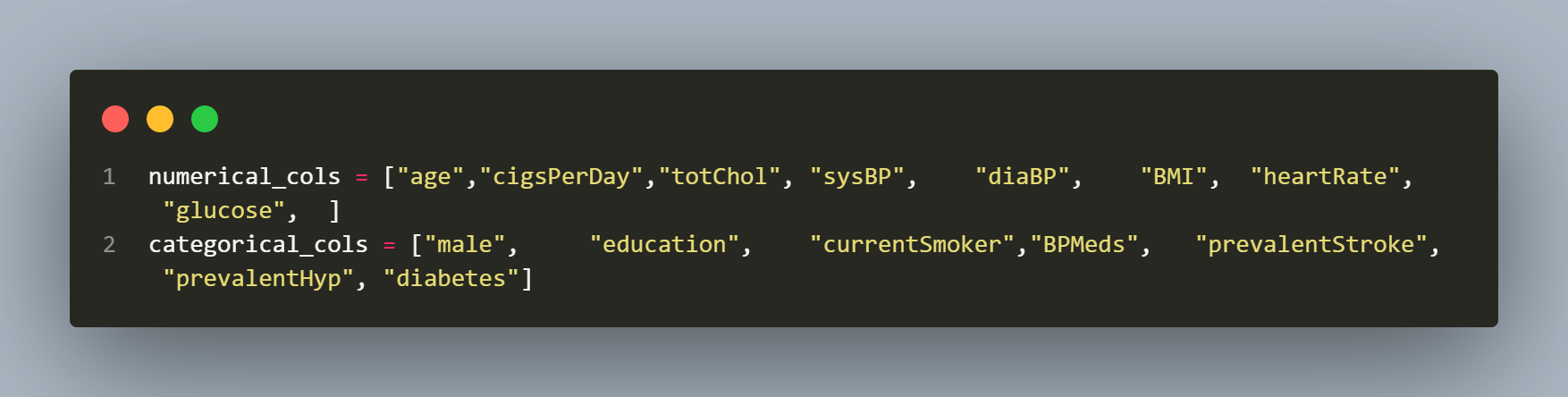
سپس به بررسی این مورد می‌پردازیم که آیا در سطرها مقادیری مانند ؟ وجود دارد یا خیر. همچنین بررسی می‌کنیم که آیا در ستون‌های غیر مجاز، مقدار صفر وجود دارد یا خیر. مثلاً در ستونی مانند BMI مقدار صفر یک مقدار غیر مجاز است زیرا بی‌ام‌آی برابر با صفر نداریم. با بررسی این دو مورد متوجه می‌شویم که ستون‌ها مقادیر غیرمجاز ندارند.

سپس با متد drop\_duplicates() مقادیر تکراری را حذف می‌کنیم.

در مرحله بعدی نام ستون‌های عددی و ستون‌های دسته‌ای را در دو لیست جدا قرار می‌دهیم. بر خلاف سوال قبلی نمی‌توانیم از دیتاتایپ استفاده کنیم چون که داده‌های دسته‌ای به شکل رشته نیستند و به صورت عددی هستند. شکل (48) این تقسیم بندی را نمایش می‌دهد.



شکل 47: تعداد مقادیر خالی در هر ستون



شکل 48: ذخیره نام ستون‌ها در دو لیست مجزا

## 1-2-پر کردن مقادیر خالی

ابتدا دیتاست را به دو دسته آموزشی و تستی تقسیم می‌کنیم. برای پر کردن مقادیر خالی ستون‌های عددی، از دو روش استفاده می‌کنیم. پر کردن مقادیر خالی با استفاده از KNN و پر کردن مقادیر با استفاده از میانه. برای پر کردن مقادیر خالی ستون‌های دسته‌ای از مد استفاده می‌کنیم. به این منظور دو شی از کلاس Pipeline ایجاد می‌کنیم که کارکرد آن، اجرای یک یا چند عملیات به طور متوالی است. پایپ‌لاین ایجاد شده برای داده‌های عددی به این صورت است که ابتدا مقادیر را استاندارد سازی می‌کنیم، سپس با استفاده از KNN مقادیر خالی را پر می‌کنیم. شکل (49) این پایپ‌لاین را نمایش می‌دهد.



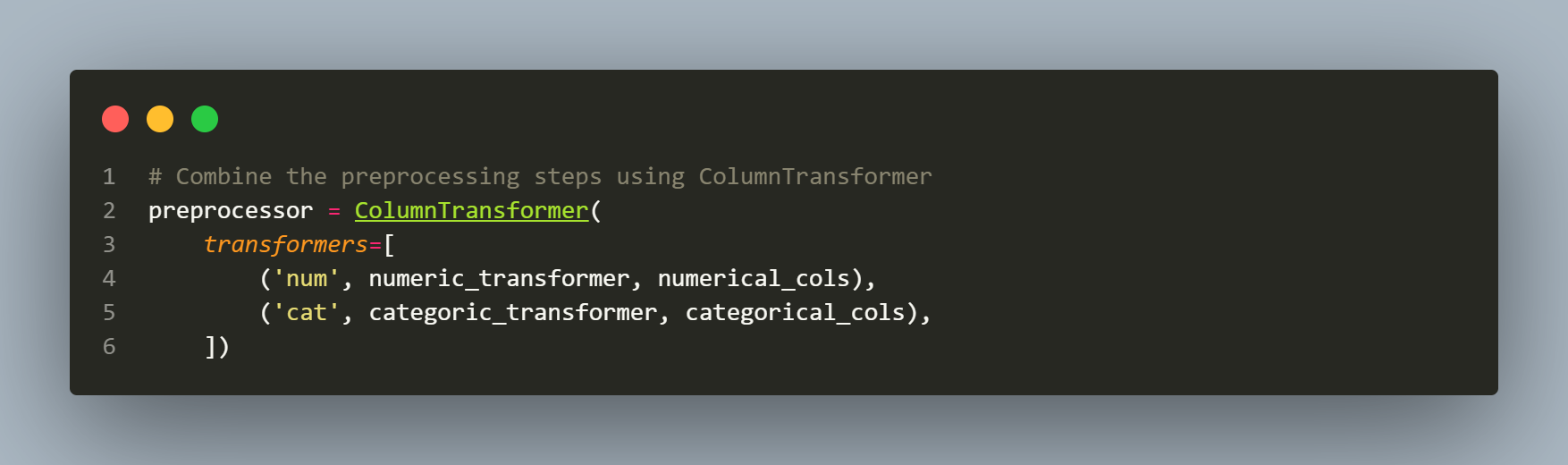
شکل 49: ایجاد پایپ‌لاین برای داده‌های عددی

پایپ‌لاین مقادیر دسته‌ای به این صورت است که با استفاده از کلاس SimpleImputer و آرگومان strategy = ‘most\_frequent’ مقادیر خالی را با استفاده از مد پر می‌کنیم و سپس داده‌ها را استانداردسازی می‌کنیم. شکل (50) پایپ‌لاین داده‌های دسته‌ای را نمایش می‌دهد.



شکل 50: پایپ‌لاین برای مقادیر دسته‌ای

سپس یک شی از کلاس ColumnTransformer ایجاد می‌کنیم. و به آرگومان transformers لیستی شامل دو تاپل را پاس می‌دهیم. هر تاپل به ترتیب شامل یک اسم، شی پایپ‌لاین و لیست ستون‌های متناظر با پایپ‌لاین است. این شی عملیات‌های تعریف شده در پایپ‌لاین‌های بالا را ستون به ستون روی لیست ستون‌های داده‌ شده اجرا می‌کند. شکل (51) این شی را نمایش می‌دهد.



شکل 51: شی ساخته شده از ColumnTransformer

حال از یکی شی دیگر از کلاس پایپ‌لاین استفاده می‌کنیم تا شی preprocessor و مدل جنگل تصادفی را به آن می‌دهیم. مزیت استفاده از این روش این است که می‌توان چندین عملیات را به طور زنجیره‌وار برای اجرا شدن تعریف کنیم و از اجرای بی‌نقص آنها اطمینان حاصل کنیم. همچنین کد نوشته شده با این روش بسیار خواناست و همچنین عوض کردن آن برای امتحان و اجرای روش‌های مختلف به راحتی صورت می‌گیرد. شکل (52) پایپ‌لاین نهایی را نمایش می‌دهد.



شکل 52: پایپ‌لاین نهایی

این پایپ‌لاین به شرح زیر عمل می‌کند:

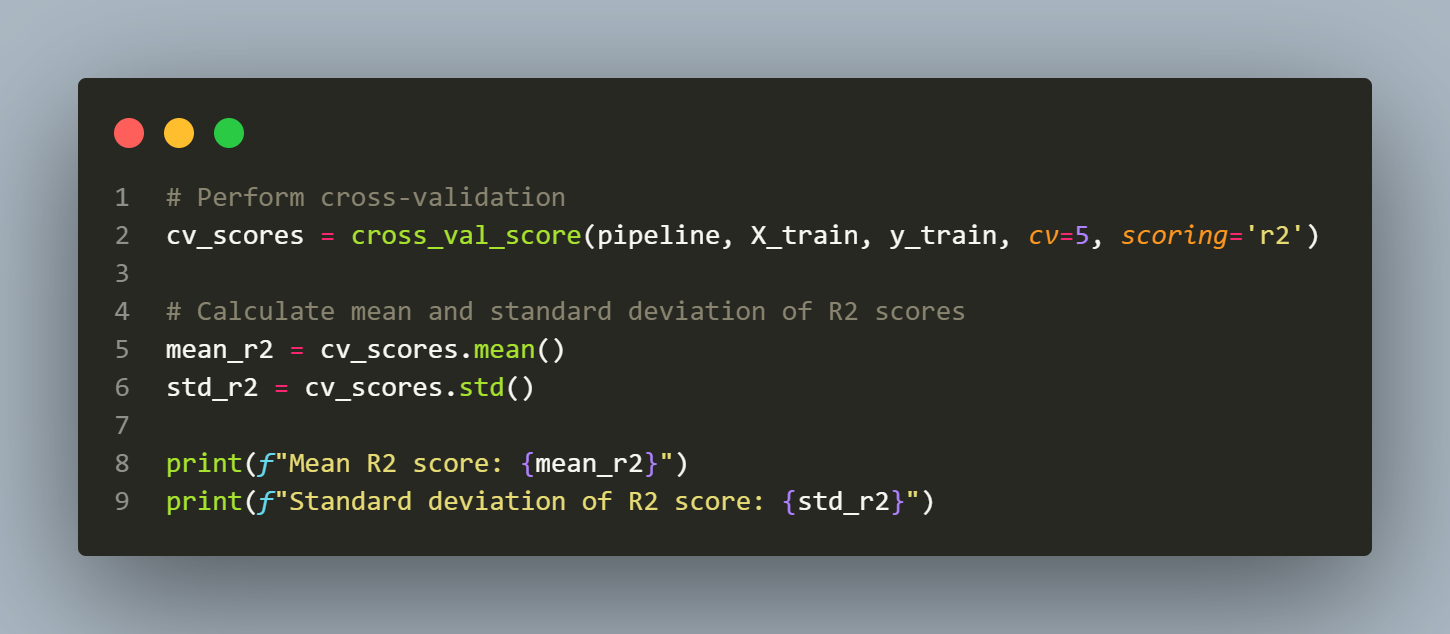
1-مقادیر خالی ستون‌های عددی را با KNN پر می‌کند و سپس داده‌ها را استاندارد می‌کند.

2-مقادیر خالی ستون‌های دسته‌ای را با مد پر می‌کند و داده‌ها را استاندارد می‌کند

3-بسته به متدهایی که در ادامه استفاده می‌شود، مدل را آموزش می‌دهد یا تست می‌کند.

برای ارزیابی روش انتخاب شده، از Cross Validation استفاده می‌کنیم. برای این منظور از کلاس cross\_val\_score یک شی می‌سازیم. به این شی، پایپ‌لاین نهایی و داده‌های آموزشی ، روش ارزیابی و تعداد بخش‌ها را پاس می‌دهیم. این بخش از کد، داده‌ها را به پنج دسته تقسیم می‌کند، عملیات‌های گفته شده را روی داده‌های آموزش و ارزیابی انجام می‌دهد و مقدار R2 را محاسبه می‌کند.

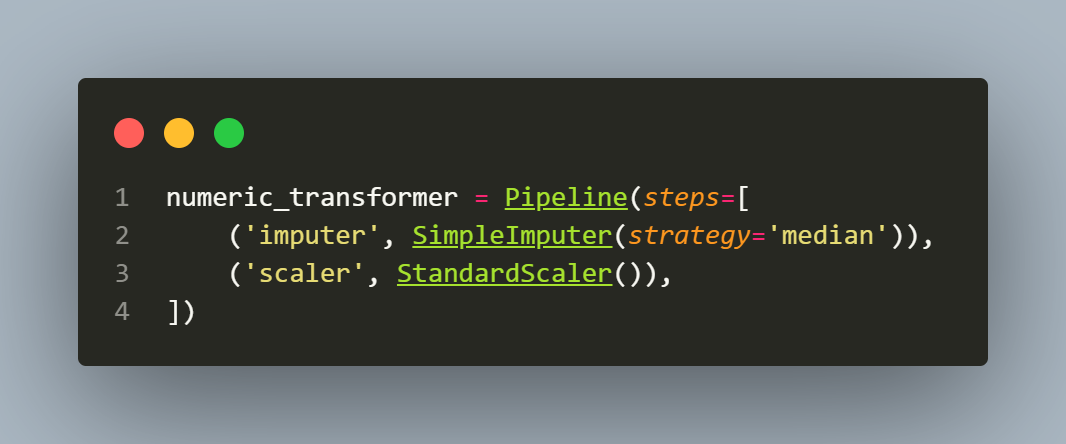
سپس میانگین و واریانس مقادیر R2 را محاسبه می‌کنیم و نمایش می‌دهیم. شکل (53) نحوه کارکرد این بخش را نمایش می‌دهد.



شکل 53: انجام Cross Validation و محاسبه R2

با این روش مقدار R2 برابر با 0.2- به دست می‌آید.

حال همین متد را برای پر کردن مقادیر با میانه استفاده می‌کنیم. با این تفاوت که در پایپ‌لاین مربوط به پیش‌پردازش داده‌های عددی، به جای کلاس KNNImputer() از کلاس SimpleImputer با آرگومان strategy = ‘median’ استفاده می‌کنیم. شکل (54) پایپ‌لاین به روز شده را نمایش می‌دهد.



شکل 54: پایپ‌لاین به روز شده برای مقادیر عددی

حال تمامی مراحل را مجددا تکرار می‌کنیم. با روش جدید، مقدار R2 برابر با 0.21- می‌شود.

روش KNN برای پر کردن مقادیر گمشده عملکرد بهتری دارد.

## 2-2-استفاده از SelectKBest

**این روش چگونه کار می‌کند؟** SelectKBest یک روش انتخاب ویژگی از scikit-learn است که k ویژگی برتر را بر اساس بالاترین امتیاز از یک آزمون آماری بین هر ویژگی و متغیر هدف انتخاب می‌کند. هنگامی که SelectKBest اعمال می شود، ابتدا امتیاز هر ویژگی را با استفاده از تابع امتیازدهی ارائه شده، مانند chi2، f\_classif، یا mutual\_info\_classif برای وظایف طبقه بندی، یا f\_regression برای وظایف رگرسیون محاسبه می کند. این امتیازات وابستگی بین هر ویژگی و هدف را اندازه گیری می کند و به SelectKBest اجازه می دهد تا ویژگی ها را رتبه بندی کند. پس از رتبه‌بندی، فقط k ویژگی‌های دارای بالاترین امتیاز حفظ می‌شوند و بقیه کنار گذاشته می‌شوند.

برای استفاده از این روش، ابتدا متغیرها و ستون هدف را در دو متغیر X و y ذخیره می‌کنیم. سپس از کلاس SelectKBest() یک شی می‌سازیم. آرگومان score\_func را با کلاس chi2 از کتابخانه سایکیت‌لرن مقدار دهی می‌کنیم و آرگومان k را برابر 10 قرار می‌دهیم.

سپس با استفاده از متد fit\_transform() و پاس دادن متغیرها و سطر هدف، ده ویژگی برتر را انتخاب می‌کنیم و نتیجه را در X\_best\_features ذخیره می‌کنیم. حال با استفاده از متد .get\_support() اندیس ستون‌های انتخاب شده را در یک لیست ذخیره می‌کنیم سپس نام ستون‌های انتخاب شده را در لیست selected\_features ذخیره می‌کنیم. شکل (55) نحوه انجام این عملیات‌ها را نمایش می‌دهد.



شکل 55: انتخاب ویژگی با استفاده از SelectKBest

حال می‌توان از لیست ایجاد شده، به راحتی برای انتخاب ستون‌های دیتافریم استفاده کرد. پس از انتخاب این ستون‌ها به سراغ آموزش مدل KNN می‌رویم.

## 3-2-آموزش KNN

ابتدا دیتاست را به دو دسته آموزشی و تستی تقسیم می‌کنیم. توجه شود از آنجایی که دیتاست imbalance است، در اولین اقدام باید تعداد لیبل‌های صفر و تعداد لیبل‌های یک را در دیتاست آموزشی برابر کرد. برای این منظور از روش SMOTE استفاده می‌کنیم. این روش یکی از روش‌های oversampling است که از لیبلی که تعداد کمتری دارد داده‌های مصنوعی تولید می‌کند تا تعداد هر دو لیبل با هم برابر شود.

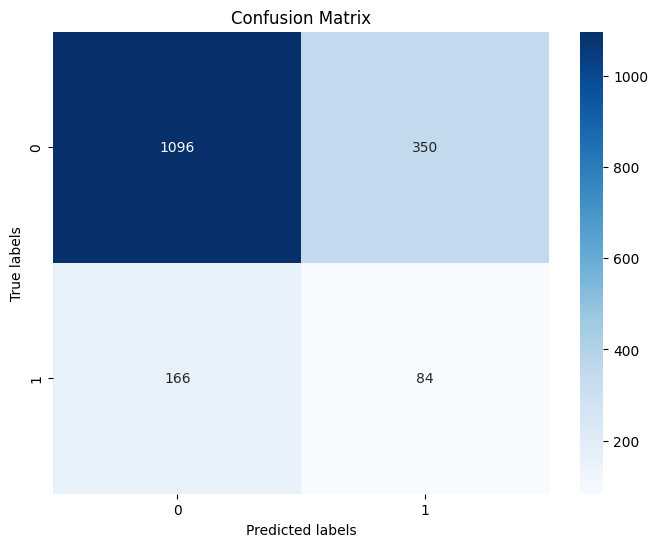
برای این منظور از کلاس SMOTE یک شی به نام sm می‌سازیم. سپس با استفاده از متد fit\_resample() داده‌های آموزشی را بالانس می‌کنیم. شکل (56) نحوه انجام این کار را نمایش می‌دهد.



شکل 56: Oversample کردن داده‌های آموزشی

حال داده‌ها برای آموزش آماده هستند. برای آموزش داده‌ها همانند بخش‌های قبلی یک پایپ‌لاین درست می‌کنیم. نحوه کارکرد این پایپ‌لاین به این صورت است که ابتدا داده‌ها را استاندارد سازی می‌کنیم سپس با استفاده از کلاس KNeighborsClassifier() و آرگومان n\_neighbors =3 یک مدل کا نزدیک‌ترین همسایه را آموزش یا تست می‌کنیم. با استفاده از متد .fit() و پاس دادن داده‌های آموزشی، ابتدا داده‌ها را استانداردسازی می‌کنیم و سپس مدل را آموزش می‌دهیم. همچنین با متد .predict() داده‌های تستی را استاندارد سازی و پیش بینی را انجام می‌دهیم. سپس با توابع r2\_score و accuracy\_score امتیاز R2 و دقت را محاسبه می‌کنیم. در این حالت، این دو مقدار به ترتیب 1.42- و 69.57 درصد به دست می‌آید.

سپس با تابع confusion\_matrix ماتریس درهم‌ریختگی را محاسبه می‌کنیم و با استفاده از تابع heatmap() آن را رسم می‌کنیم. شکل 57 ماتریس درهم ریختگی به دست آمده را نمایش می‌دهد.



شکل 57: ماتریس درهم ریختگی مدل آموزش داده شده

## 4-2-آموزش و تست مدل با فاصله‌های متفاوت

فاصله‌ی پیش‌فرض مدل KNN فاصله اقلیدسی است. حال با فواصل مختلف، آموزش و تست را تکرار می‌کنیم.

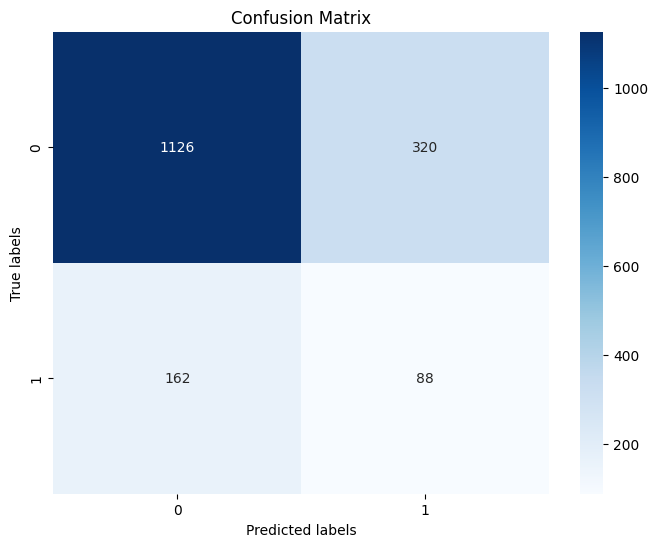
### 1-4-2-فاصله منهتن

پایپ‌لاین آموزش را به این صورت تغییر می‌دهیم که آرگومان metric را برابر ‘manhattan’ قرار می‌دهیم. شکل (58) پایپ‌لاین به روز شده را نمایش می‌دهد.



شکل 58: پایپ‌لاین آموزش/تست با فاصله متهتن

سپس همانند قبل مراحل آموزش و تست و ارزیابی را تکرار می‌کنیم. در این حالت مقدار R2 برابر با 1.26- و دقت برابر با 0.7158 به دست می‌آید. مشاهده می‌شود که نتایج بهبود داشته‌اند. همچنین شکل (59) ماتریس درهم‌ریختگی حالت جدید را نمایش می‌دهد.



شکل 59: ماتریس درهم ریختگی KNN با فاصله منهتن

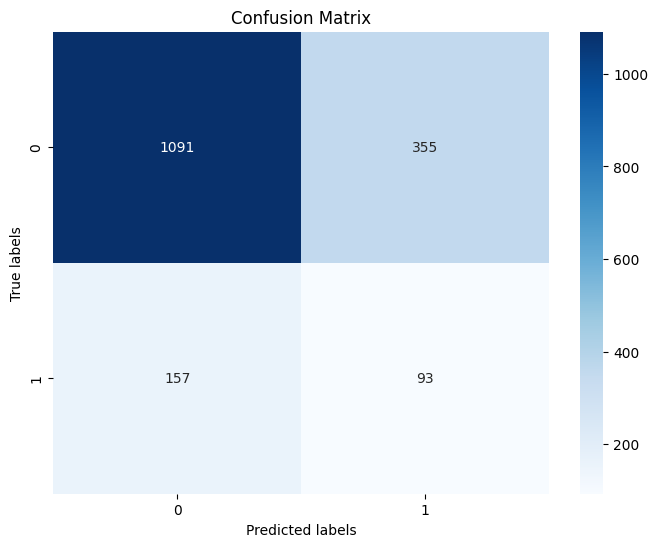
### 2-4-2-فاصله کسینوسی

پایپ‌لاین آموزش را به این صورت تغییر می‌دهیم که آرگومان metric را برابر ‘cosine’ قرار می‌دهیم. شکل (60) پایپ‌لاین به روز شده را نمایش می‌دهد.



شکل 60: پایپ‌لاین تست و آموزش KNN با فاصله کسینوسی

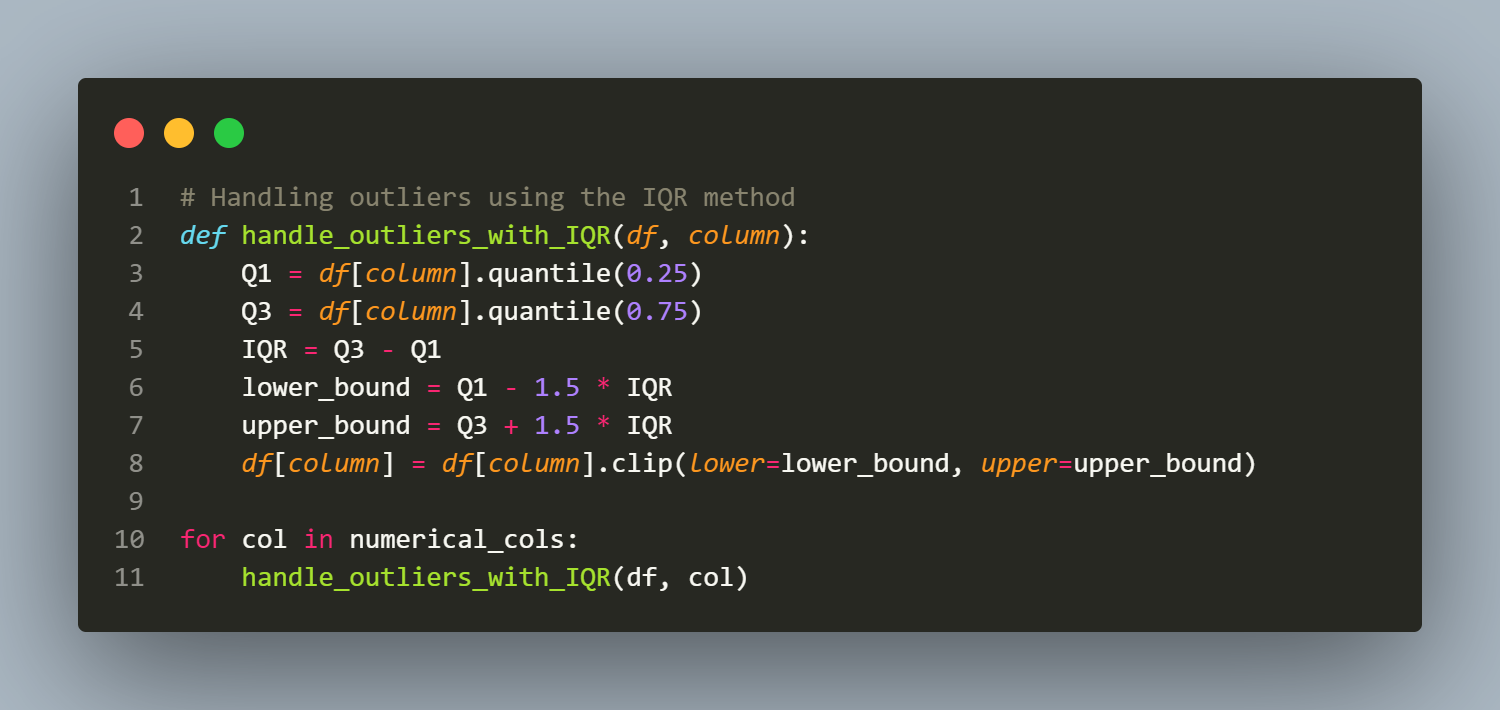
مراحل تست و آموز ش را تکرار می‌کنیم. در این حالت R2 برابر با 1.4- و دقت برابر 0.6981 به دست می‌آیند. مشاهده می‌شود که عملکرد فاصله کسینوسی با فاصله اقلیدسی تفاوت چندانی ندارد.



شکل 61: ماتریس درهم‌ریختگی مدل KNN با فاصله کسینوسی

## 5-2-انجام پیش‌پردازش بیشتر برای رسیدن به نتایج بهتر

یکی از مراحل پیش‌پردازشی که انجام نشد، پیدا کردن و حذف داده‌های پرت بود. به این منظور. ابتدا برای هر ستون، مقادیر چارک اول و سوم و دامنه میان چارکی را پیدا می‌کنیم، سپس داده‌های بزرگتر از 1.5 برابر دامنه به علاوه چارک سوم و داده‌های کوچک‌تر از 1.5 برابر دامنه منهای چارک اول هستند را حذف می‌کنیم. برای این منظور از تابع .clip() و تعریف دو مقدار توضیح داده شده به عنوان حد بالا و حد پایین، تنها داده‌هایی را نگه می‌داریم که بین دو حد تعریف شده هستند. شکل (61) نحوه انجام این کار را نمایش می‌دهد.



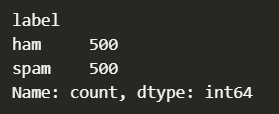
شکل 62: حذف داده‌های پرت

سپس تقسیم بندی داده‌ها به آموزش و آزمایش را انجام می‌دهیم و با استفاده از پایپ‌لاین بخش قبلی، پس از استانداردسازی داده‌ها، یک مدل KNN با تعداد همسایگی 3 و فاصله منهتن را آموزش می‌دهیم و پس از تست، ارزیابی می‌کنیم.

در این حالت مقدار R2 برابر با 0.48- و دقت برابر با 0.8136 به دست می‌آید که به وضوح تاثیر حذف داده‌ها بر عملکرد مدل را نشان می‌دهد.

# 3-پاسخ سوال سوم

پس از خواندن دیتاست، ابتدا تعداد رکورد موجود از هر لیبل را نمایش می‌دهیم. شکل (63) خروجی این کار را نمایش می‌دهد.



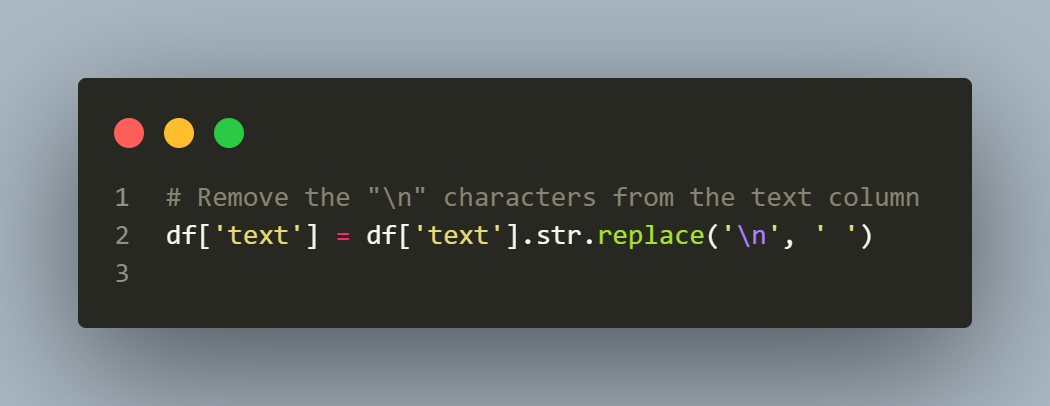
شکل 63: تعداد ایمیل‌های با لیبل سالم و اسپم

مشاهده می‌شود که دیتاست بالانس است. در ادامه لیبل‌های ham را به صفر و spam را به یک تبدیل می‌کنیم.

## 1-3-پیش پردازش داده‌ها

### 1-1-3-حذف کاراکترهای \n

اولین قدم، حذف کاراکتر \n از رشته‌ی هر ایمیل است. برای این منظور از متد .replace() استفاده می‌کنیم. کد شکل (64) نحوه انجام این کار را نمایش می‌دهد.



شکل 64: حذف کاراکتر \n

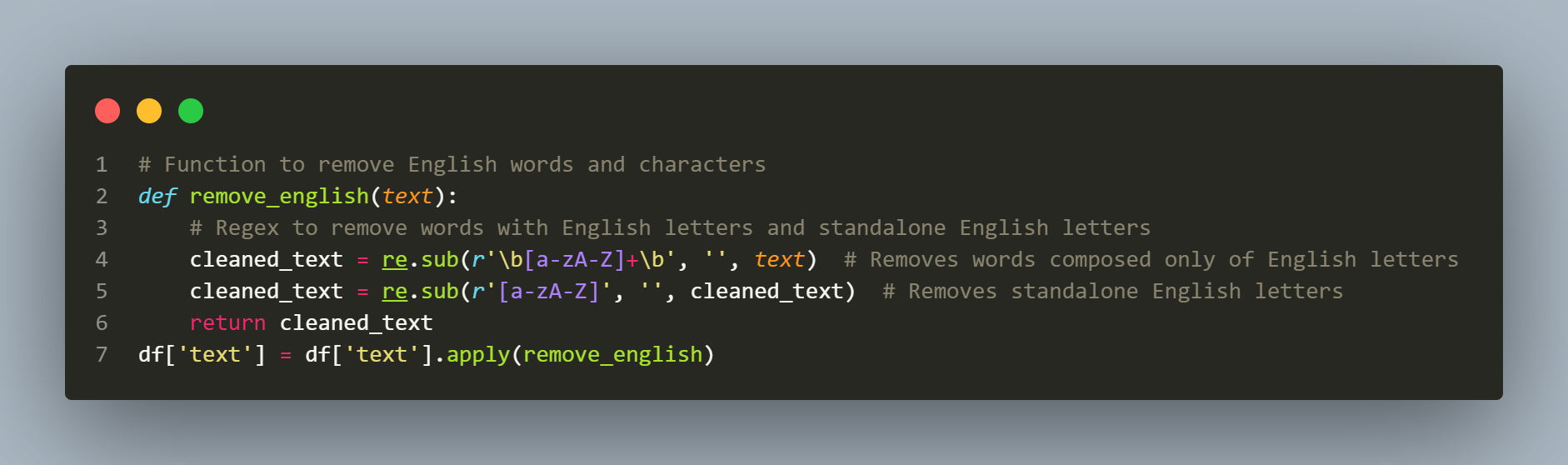
### 2-1-3-حذف کاراکترهای انگلیسی

قدم بعدی حذف کلمات و کاراکترهای انگلیسی است. دلیل انجام این کار این است که زبان ایمیل‌ها فارسی است و وجود کاراکترهای انگلیسی باعث می‌شود که مدل روی کلماتی آموزش ببیند که لزوماً اطلاعاتی برای تصمیم‌گیری مدل ارائه نمی‌دهد. همچنین پس از حذف علائم نگارشی، از مواردی مانند ایمیل یا لینک‌ها، یک رشته‌ی بلند و نه چندان معنادار از کاراکترهای فارسی می‌ماند که مدل را گمراه می‌کند.

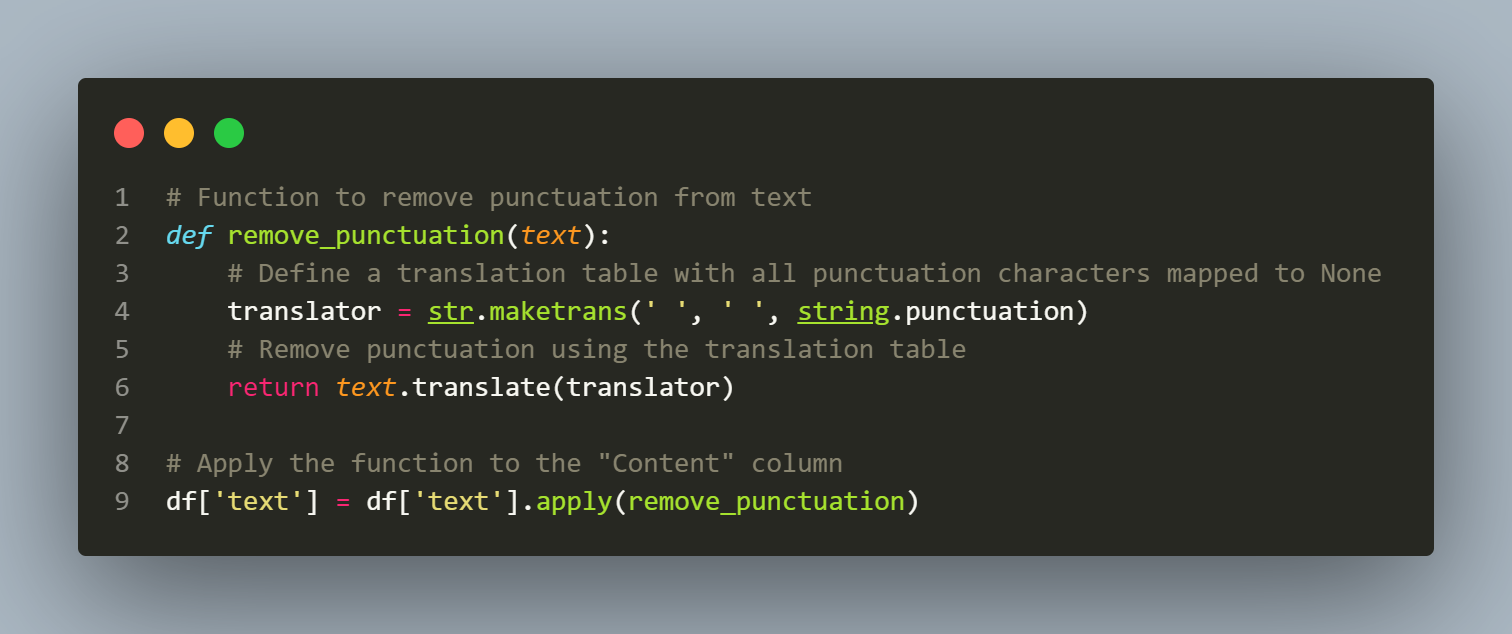
برای این منظور از کتابخانه رجکس استفاده می‌کنیم. ابتدا یک تابع را تعریف می‌کنیم که ردیف به ردیف سطرها را دریافت می‌کند و سپس با استفاده از متد sub() کاراکترهای انگلیسی را حذف می‌کند و متن تمیز شده را برمی‌گرداند. شکل (65) نحوه انجام این کار را نمایش می‌دهد.

### 3-1-3-حذف علائم نگارشی

در مرحله بعدی، علائم نگارشی را حذف می‌کنیم. دلیل انجام این کار این است که تکرار یکیسری از علائم مانند نقطه یا ات‌ساین باعث گمراه شدن مدل می‌شود در حالی که حضور این موارد تعیین کننده اسپم بودن ایمیل‌ها نیست. شکل (66) نحوه انجام این کار با متد maketrans()



شکل 65: حذف کاراکترهای انگلیسی

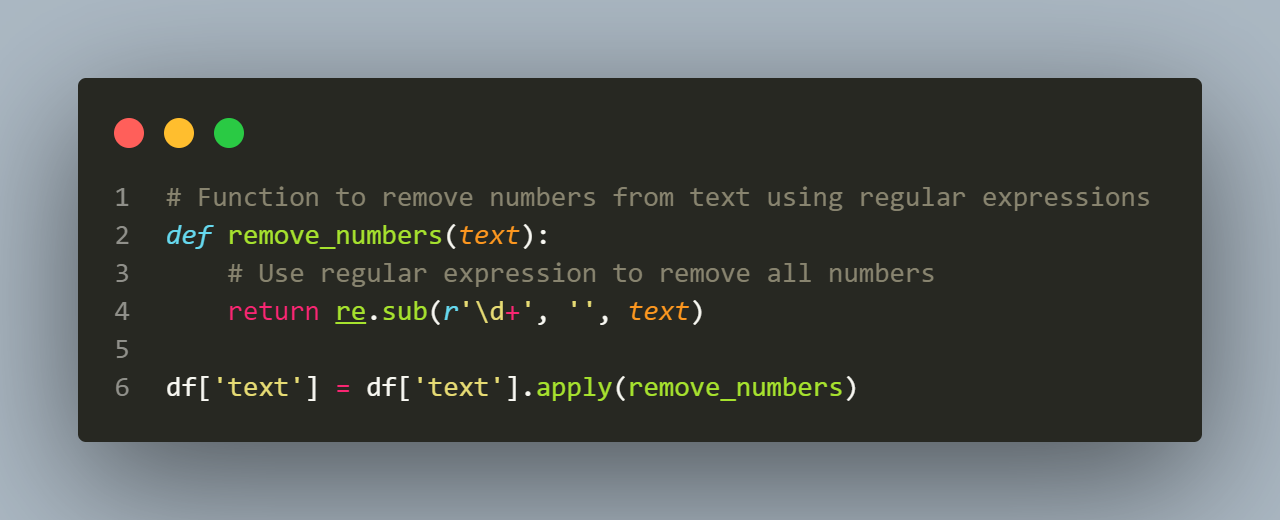


شکل 66: حذف علائم نگارشی

را نمایش می‌دهد. همچنین علائم نگارشی از اتریبیوت string.punctuation فراخوانی خواهند شد.

4-1-3-حذف اعداد

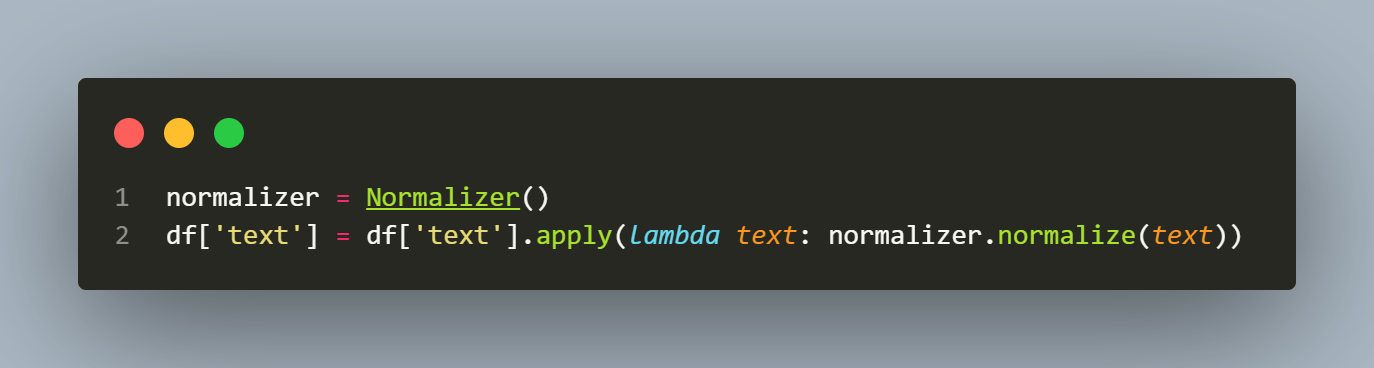
پس از انجام این مراحل، اعداد را از ایمیل‌ها حذف می‌کنیم. وجود اعداد باعث ایجاد نویز در داده‌ها می‌شود. برای حذف اعداد مجدداً همانند قبل از رجکس و تابعی استفاده می‌کنیم که با متد apply() سطر به سطر ایمیل‌ها را دریافت می‌کند و اعداد را از آنها حذف می‌کند.



شکل 67: حذف اعداد با استفاده از رجکس

### 5-1-3- نرمال کردن متن‌ها

نرمال کردن متن ریشه‌بندی، لغت‌سازی و حذف کلید واژه، ابعاد داده‌های متن را با جمع کردن کلمات مشابه در یک نمایش مشترک کاهش می‌دهد. این کار فضای ویژگی ها را ساده می‌کند و کارایی الگوریتم‌ها را بهبود می‌بخشد. برای نرمال کردن متن از کتابخانه Hazm یک شی از کلاس Normalizer() می‌سازیم. سپس روی ستون text متد apply() را اجرا می‌کنیم و یک تابع لامبدا به این متد پاس می‌دهیم که آرگومان آن متن است. سپس متد normalize() را روی متن پاس داده شده اجرا می‌کنیم و نتیجه را بر می‌گردانیم. شکل (68) نحوه انجام نرمال‌سازی را نمایش می‌دهد.

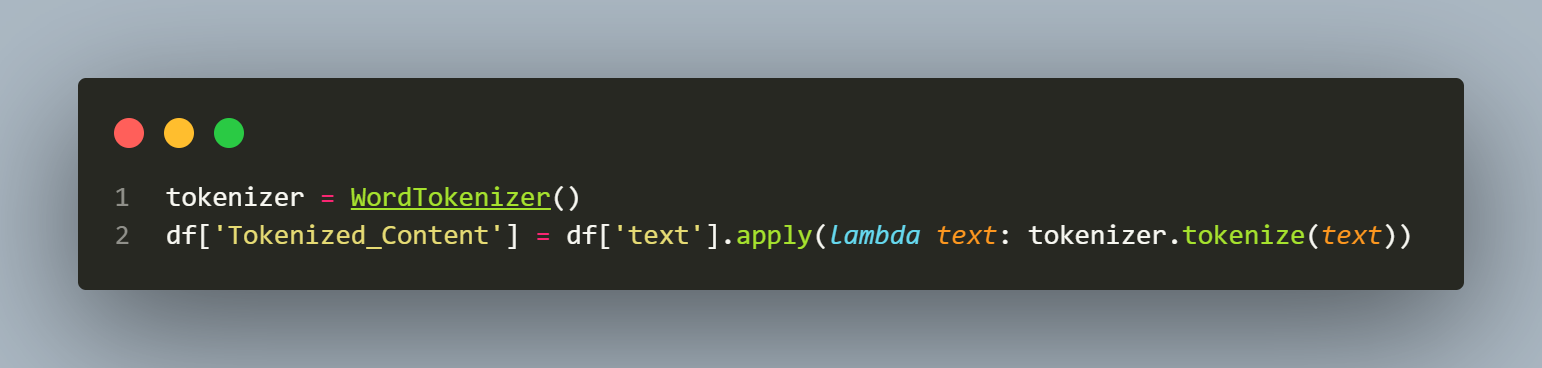


شکل 68: نحوه انجام نرمال سازی متن

### 6-1-3-توکن کردن متن

توکن‌سازی متن را به واحدهای کوچک‌تری که می توانند کلمات، عبارات یا کاراکترها باشند، تجزیه می‌کند. این جزئیات تجزیه و تحلیل دقیق‌تری از متن را امکان پذیر می کند و به الگوریتم‌ها اجازه می دهد تا بر روی واحدهای جداگانه کار کنند تا متن. توکن‌ها ساده‌ترین جز سازنده برای ایجاد ویژگی‌ها از متن هستند.

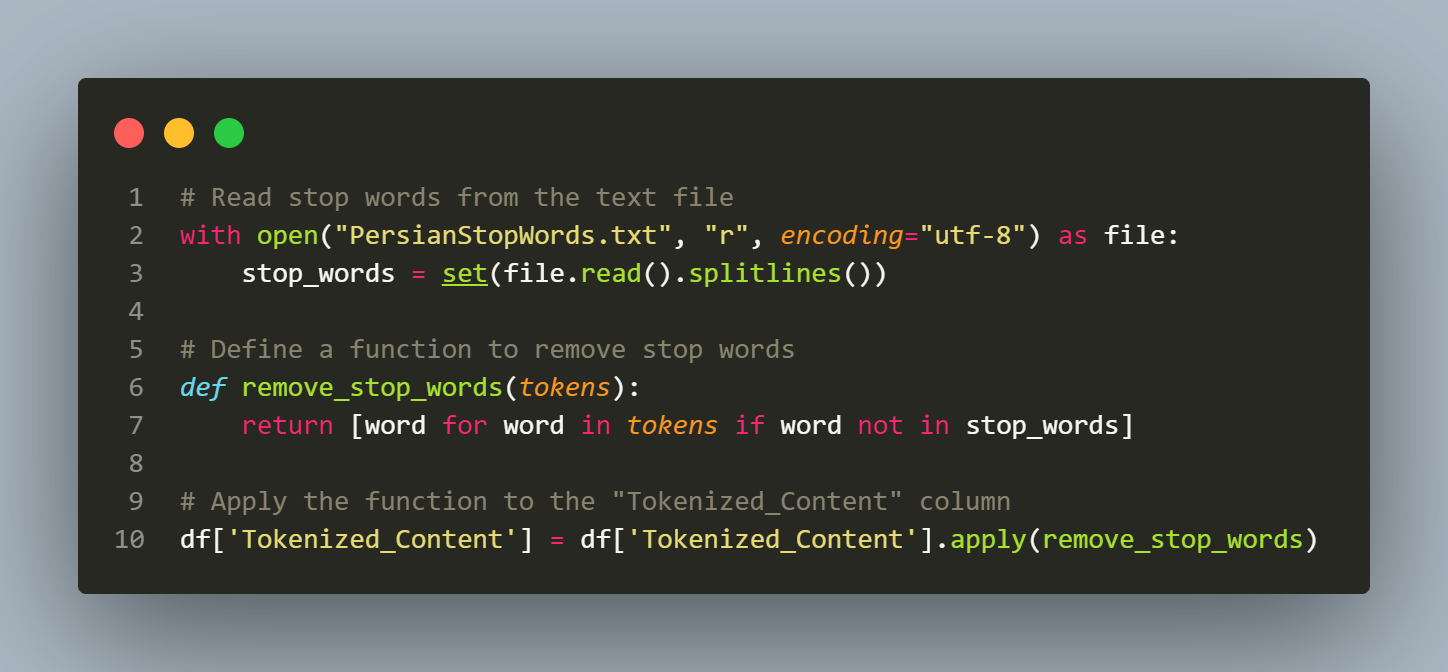
برای توکن کردن متن از کلاس WordTokenizer کتابخانه Hazm استفاده می‌کنیم. دقیقاً مانند قسمت قبلی در متد apply() یک تابع لامبدا ایجام می‌کنیم و با متد tokenize سطر به سطر متن‌ها را توکن می‌کنیم. لیست توکن‌های هر سطر را در یک ستون جداگانه به نام Tokenized\_Content ذخیره می‌کنیم. شکل (69) نحوه انجام این کار را نمایش می‌دهد.



شکل 69: توکن سازی متن

### 7-1-3-حذف ایست‌واژه‌ها (کلمات پالایشی یا stop words)

برای حذف ایست‌واژه‌ها، ابتدا این واژه‌ها را از فایل PersianStopWords.txt می‌خوانیم و در متغیر stop\_words ذخیره می‌کنیم. سپس لیست‌های موجود در ستون Tokenized\_Content را با متد apply یک به یک می‌خوانیم. سپس با استفاده از list comprehension توکن ها را یک به یک بررسی می‌کنیم که آیا در لیست ایست‌واژه‌ها موجود هستند یا نه. سپس توکن‌هایی که در ایست‌واژه‌ها موجود هستند را حذف می‌کنیم. شکل 70 نحوه انجام این کار را نمایش می‌دهد.



شکل 70: پیدا کردن و حذف ایست‌واژه‌ها

سپس توکن‌ها را در کنار هم قرار می‌دهیم و متن‌های بلند را مجددا تشکیل می‌دهیم. (به اصطلاح flatten کردن) شکل (71) نحوه انجام این کار را نمایش می‌دهد.



شکل 71: بازتولید متن‌ها از کلمه‌ها

8-1-3- استفاده از TF-IDF

TF-IDF به هر عبارت در یک سند بر اساس فراوانی آن در سند (TF) و نادر بودن آن در همه اسناد (IDF) وزن اختصاص می دهد. این طرح وزن دهی به اولویت بندی عباراتی که هم در یک سند متداول هستند و هم در کل مجموعه منحصر به فرد هستند، کمک می کند و آنها را برای کارهای پایین دستی مانند طبقه بندی یا خوشه بندی متمایزتر و آموزنده‌تر می کند.

برای استفاده از TF-IDF از کلاس TfidfVectorizer() یک شی می‌سازیم. سپس با استفاده از متد fit\_transform() و پاس دادن ستون متن بردارها را ایجاد می‌کنیم و سپس نتیجه را در یک متغیر دیگر ذخیره می‌کنیم. سپس ماتریس به دست آمده را به دیتافریم تبدیل می‌کنیم. شکل (72) نحوه انجام این کار را نمایش می‌دهد.

حال دیتاست برای آموزش KNN آماده است. این دیتاست هزار سطر و 18886 ستون دارد.

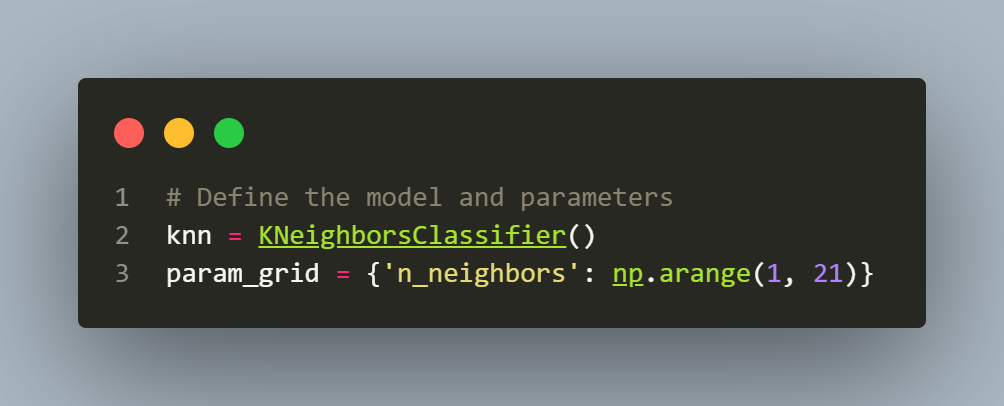
## 2-3-آموزش KNN و پیدا کردن بهترین K

ابتدا دیتاست را به دو دسته آموزشی و آزمایشی تقسیم می‌کنیم. ابتدا داده‌ها را استاندارد سازی می‌کنیم. در مرحله بعدی با استفاده از Oversampling تعداد سطرهای با لیبل‌ صفر و تعداد لیبل‌های یک را با هم برابر می‌کنیم. برای این منظور از روش SMOTE استفاده می‌کنیم. برای این منظور از کلاس SMOTE یک شی می‌سازیم و با پاس دادن داده‌های آموزشی به متد fit\_resample() کار oversampling را انجام می‌دهیم.



شکل 72: استفاده از روش TF-IDF

سپس از کلاس KNeighborsClassifier() یک شی می‌سازیم. همچنین یک دیکشنری ایجاد می‌کنیم و پارامترهایی که قرار است روی آنها Grid search انجام بدهیم را به عنوان کلید و مقادیر پارامترها را به عنوان value در دیکشنری قرار می‌دهیم. شکل (73) تعریف شی و این دیکشنری را نمایش می‌دهد.



شکل 73: تعریف مدل و دیکشنری پارامترها

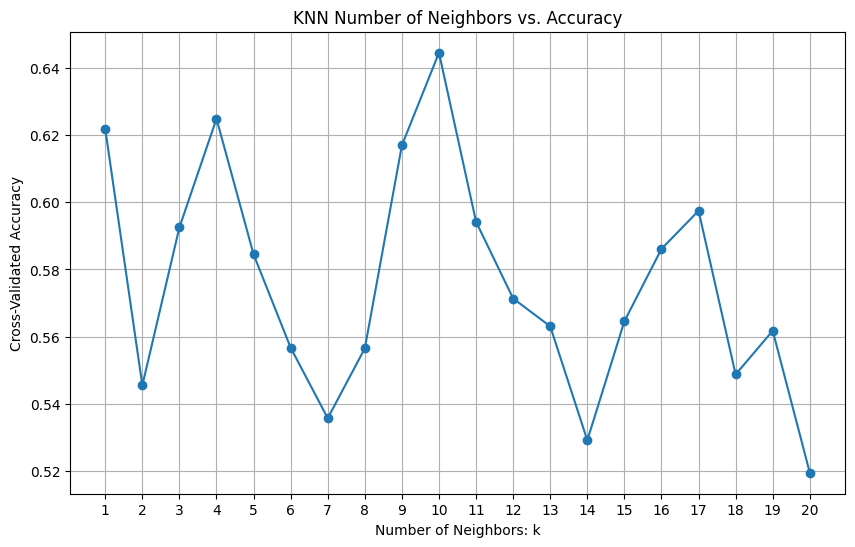
سپس از کلاس GridSearchCV() یک شی می‌سازیم. آرگومان‌های پاس داده شده به این کلاس، به ترتیب مدل تعریف شده، دیکشنری پارامترها، تعداد بخش‌ها برای CrossValidation و معیار انتخاب شده برای ارزیابی است. تعداد بخش‌ها را پنج قرار می‌دهیم و معیار ارزیابی را دقت قرار می‌دهیم. سپس با متد .fit() و پاس دادن داده‌های آموزشی، Grid search را آغاز می‌کنیم.

شی ساخته شده از GridSearchCV() هر بار پارامتر n\_neighbors را تنظیم می‌کند، CV را انجام می‌دهد و در نهایت یک مقدار برای دقت مدل باز می‌گرداند. حال می‌توان از این مقادیر، یک نمودار دقت بر اساس K رسم کرد.

شکل (74) کدی که این نمودار را رسم می‌کند نمایش می‌دهد. همچنین شکل (76) خروجی این نمودار را نمایش می‌دهد. با توجه مشاهده می‌شود که K بهینه برابر با ده است و در این حالت دقت برابر با 0.6444 خواهد شد.



شکل 74: کد رسم نمودار دقت بر اساس K



شکل 75: نمودار دقت بر اساس K