

**دانشکده ریاضی**

**گروه علوم کامپیوتر و داده**

**گزارش تمرین شماره دو**

**درس داده کاوی**

**نام دانشجو**

**محمد گرامی فر**

اردیبهشت ماه 1401

فهرست گزارش

[دیتاست شماره ۱ 2](#_Toc103896049)

[1) 5](#_Toc103896050)

[2) 8](#_Toc103896051)

[3) 9](#_Toc103896052)

[4) 9](#_Toc103896053)

[5) 10](#_Toc103896054)

[6) و 7) 11](#_Toc103896055)

[**8)** 14](#_Toc103896056)

[۹ (آ) 15](#_Toc103896057)

[۹ (ب) و ۱۰ 16](#_Toc103896058)

[۹ (ج) 19](#_Toc103896059)

[۹ (د) و ۱۰ 23](#_Toc103896060)

[11) 26](#_Toc103896061)

[12) و 13) 28](#_Toc103896062)

[14) 31](#_Toc103896063)

[15) 32](#_Toc103896064)

[16) 33](#_Toc103896065)

[17) 34](#_Toc103896066)

[دیتاست شماره ۲ 36](#_Toc103896067)

[1) 36](#_Toc103896068)

[2) 36](#_Toc103896069)

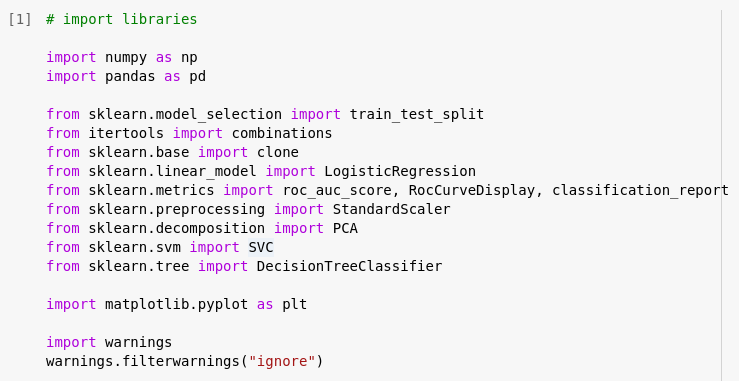
[3) 41](#_Toc103896070)

[4) 42](#_Toc103896071)

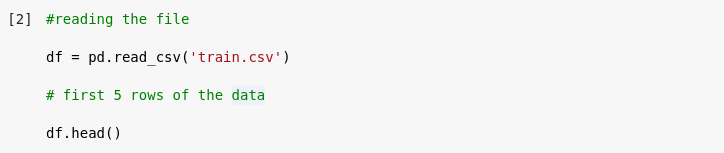
[5) 42](#_Toc103896072)

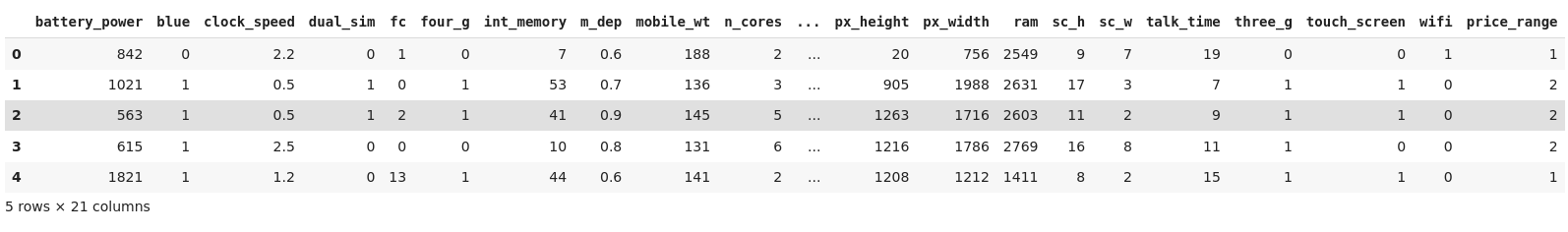
## دیتاست شماره ۱

در ابتدا کتابخانه‌هایی که نیاز داریم را وارد می‌کنیم:

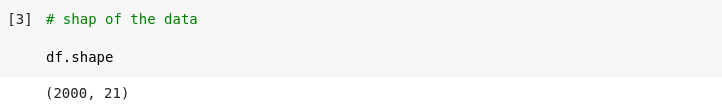


ابتدا فایل دیتا را با استفاده از کتابخانه pandas بارگذاری می‌کنیم:

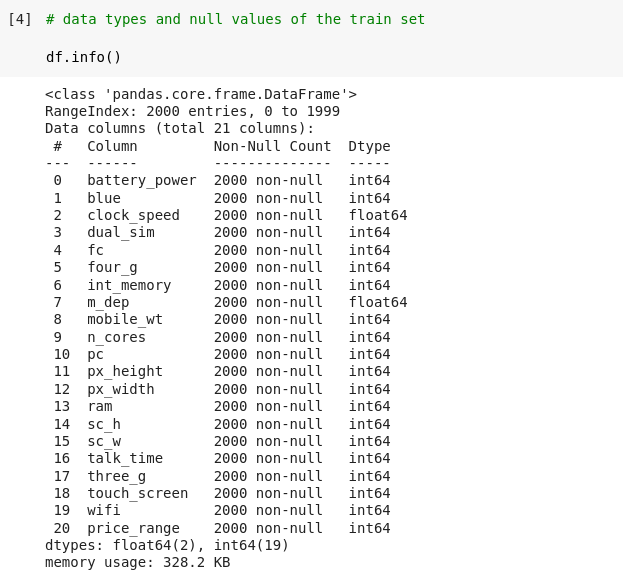




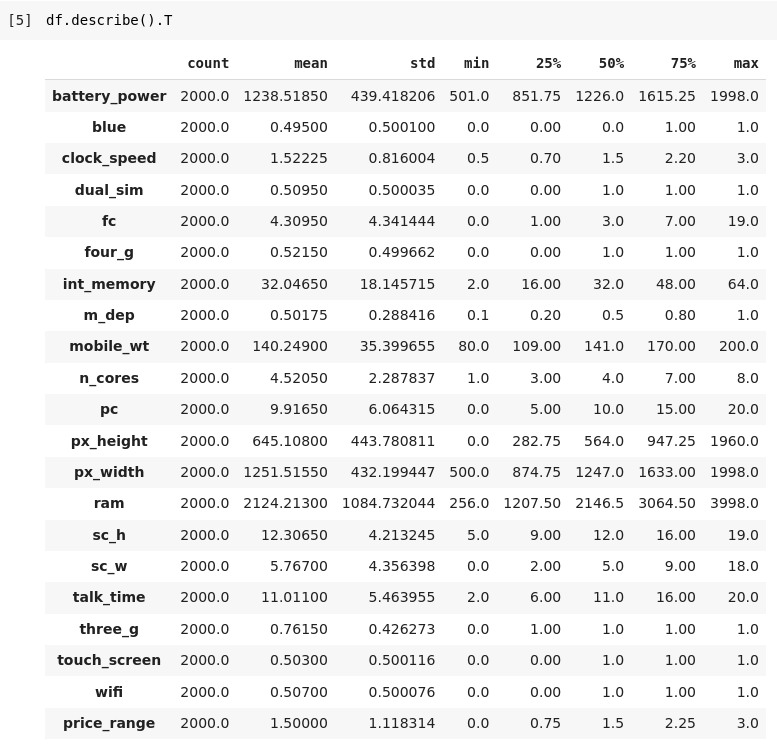
به شکل دیتاست نگاهی می‌اندازیم:



با استفاده از متد info به نوع دیتا‌ها را بررسی می‌نماییم. همچنین اگر در ستونی مقادیر از دست رفته وجود داشته باشد می‌توانیم آن را مشاهده کنیم:



با استفاده از متد describe شاخصه‌های آماری دیتاست را بررسی می‌کنیم:

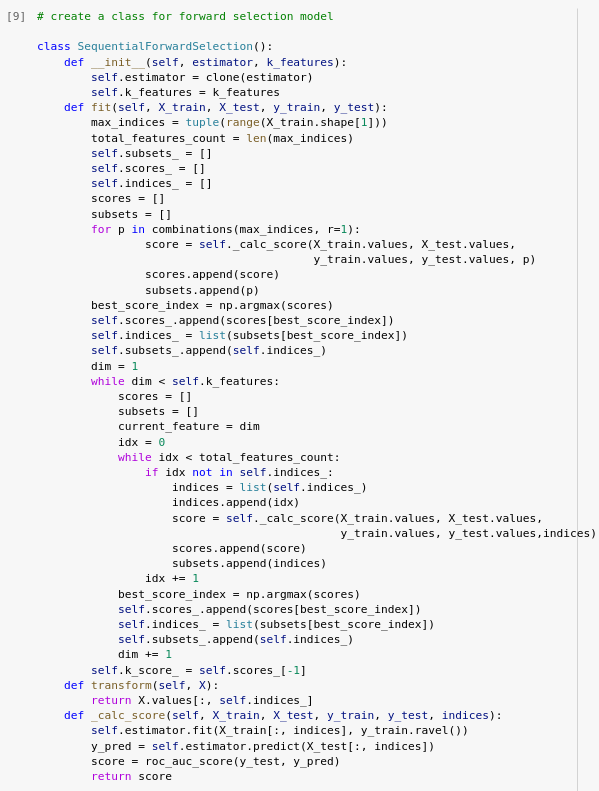


طبق دستور مسئله، محدوده قیمت گوشی‌ها را به دو دسته تبدیل می‌کنیم. برای این کار ابتدا مقادیر یکتا ستون price\_range را به دست می‌آوریم و نهایتا آن‌ها را به دو دسته کاهش می‌دهیم:

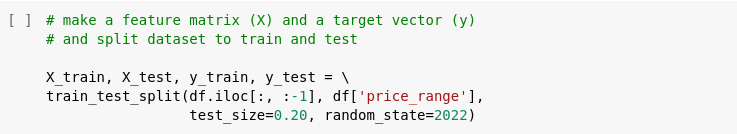


# 1)

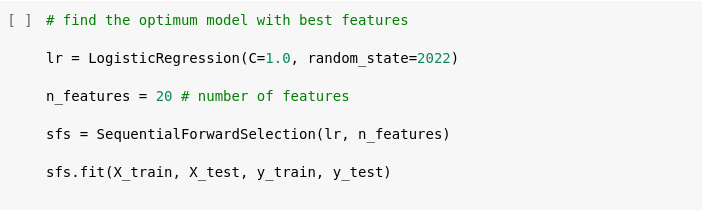
کلاس نوشته شده در پایین با گرفتن مدل یادگیری ماشین و تعداد ویژگی‌ها، با استفاده از روش Forward Selection بهترین مجموعه‌ی ویژگی‌ها را برمی‌گزیند.



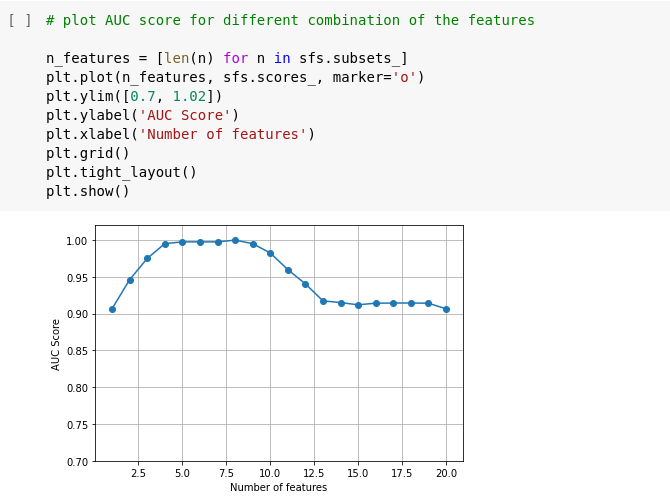
دیتاست را به فضای ویژگی‌ها و بردار هدف برای train و test تقسیم می‌کنیم:



مدل Logistic Regression را وارد Forward Selection می‌کنیم تا مجموعه بهترین ویژگی‌ها را بیابیم:

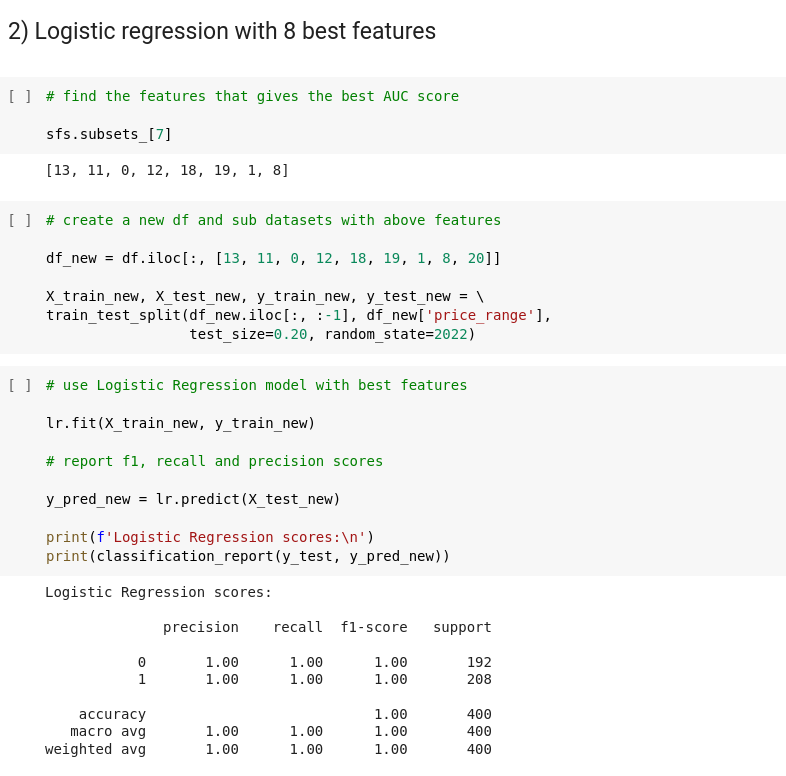


با مشاهده نمودار زیر پی می‌بریم که بهترین تعداد ویژگی‌ها، ۸ عدد از آن‌هاست:



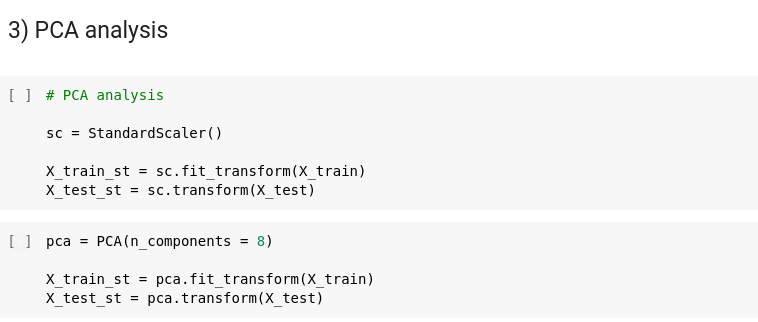
# 2)

در این مرحله مدل Logistic Regression را بر روی ۸ ویژگی برتر اعمال می‌کنیم و معیار‌های f1، recall و precision را برای آن به دست می‌آوریم



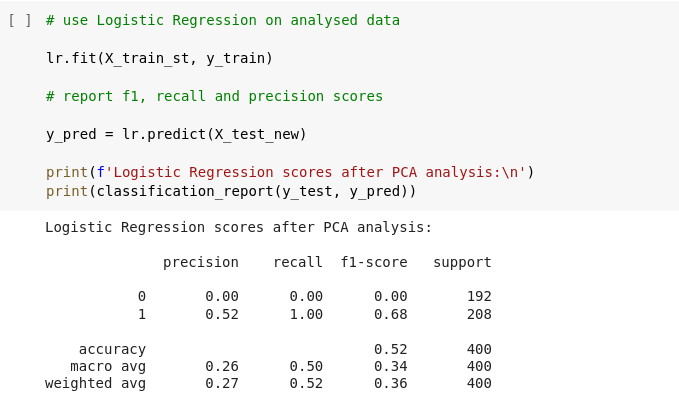
# 3)

با استفاده از الگوریتم PCA تعداد ویژگی‌ها را به ۸ عدد کاهش می‌دهیم



# 4)

الگوریتم Logistic Regression را بر روی دیتاست با ویژگی‌های کاهش یافته پیاده می‌کنیم و معیار‌های خواسته شده را برای آن بدست می‌آوریم



# 5)

در الگوریتم ماشین بردار پشتیبان، داشتن یک خط راست بین این دو کلاس آسان است. وقتی داده ها خطی جدایی پذیر نباشند، باید از کلاس بندیهایی با مرزهای تصمیم گیری غیرخطی برای کلاس بندی داده ها استفاده کنیم. یکی از راه های کلاس بندی چنین داده هایی استفاده از توان دوم، سوم یا چندم متغیرهای ورودی است. ولی زمانی که وقتی از SVM استفاده می‌کنید، می‌توانیم از یک تکنیک ریاضی با نام Kernel Trick استفاده کنید. استفاده از کرنل باعث میشود بدون اضافه کردن Polynomial Feature ، همان نتیجه‌ای رو بگیریم که انگار آن‌هارو اضافه کرده ایم. در این روش در واقع توابعی وجود دارند که فضای ورودی بُعد پایین را دریافت کرده و آن را به فضای بُعد بالاتر تبدیل می‌کنند. این تبدیل، یک مسئله غیر قابل جداسازی را به مسئله قابل جداسازی مبدل می‌کند. به این توابع، تابع‌های هسته (کرنل) گفته می‌شود.

از آنجایی که ما می خواهیم نمونه ها به صورت خطی در فضای ویژگی قابل تفکیک باشند، کیفیت فضای ویژگی برای عملکرد ماشینهای بردار پشتیبان حیاتی است. با این حال، ما نمیدانیم کدام توابع هسته خوب هستند، چرا که ما نگاشت ویژگی را نمیدانیم. بنابراین، انتخاب هسته بزرگترین عدم قطعیت ماشینهای بردار پشتیبان است. یک هسته ضعیف نمونه ها را به یک فضای ویژگی ضعیف نگاشت میکند و در نتیجه عملکرد ضعیفی دارد. به عبارت دیگر، نوع تابع هسته برای یک مساله معین از داده ها یاد گرفته نمیشود و باید آن را مشخص ک نیم. از اینرو، انتخاب تابع هسته یک ابرپارامتر است. برای مثال هر کدام از توابع کرنل در بعضی زمینه ها بیشتر کاربرد دارند:

**کرنل چند جمله ای**

این کرنل در پردازش تصویر پرکاربرد است. معادله آن به صورت زیر است :

فرمول کرنل چند جمله ای

**کرنل گاوسی**

این یک کرنل برای اهداف عمومی است. و هنگامی که هیچ دانش پیشینی در مورد داده ها وجود ندارد استفاده می شود. معادله آن به صورت زیر است :

فرمول کرنل گاوسی

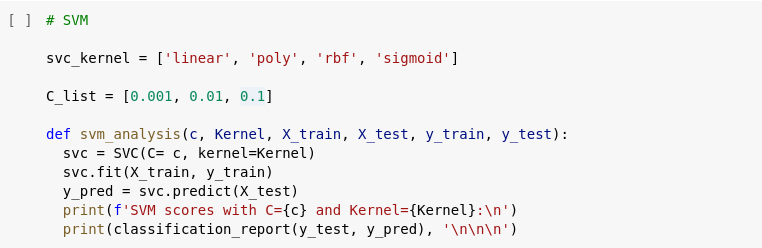
**تابع پایه شعاعی گاوسی ( RBF )**

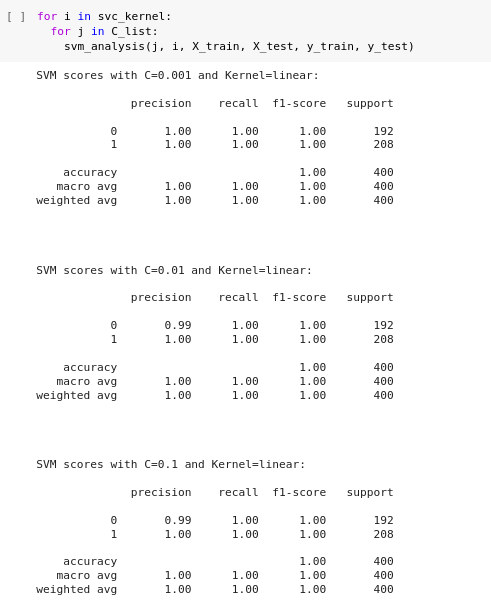
این کرنلی برای اهداف عمومی کاربرد دارد. و هنگامی که هیچ دانش پیشینی در مورد داده ها وجود نداشته باشد، مورد استفاده قرار می گیرد. معادله آن به صورت زیر است :

فرمول کرنل RBF

# 6) و 7)

بر روی دیتاست اصلی، مدل SVM را با پارامتر‌های مختلف محاسبه می‌کنیم. پارامتر‌هایی که تغییر داده می‌شوند ضریب c مدل و کرنل آن است. برای ۱۲ مدل SVM دقت‌های خواسته شده را هم چاپ می‌کنیم.





SVM scores with C=0.001 and Kernel=linear:

precision recall f1-score support

0 1.00 1.00 1.00 192

1 1.00 1.00 1.00 208

accuracy 1.00 400

macro avg 1.00 1.00 1.00 400

weighted avg 1.00 1.00 1.00 400

SVM scores with C=0.01 and Kernel=linear:

precision recall f1-score support

0 0.99 1.00 1.00 192

1 1.00 1.00 1.00 208

accuracy 1.00 400

macro avg 1.00 1.00 1.00 400

weighted avg 1.00 1.00 1.00 400

SVM scores with C=0.1 and Kernel=linear:

precision recall f1-score support

0 0.99 1.00 1.00 192

1 1.00 1.00 1.00 208

accuracy 1.00 400

macro avg 1.00 1.00 1.00 400

weighted avg 1.00 1.00 1.00 400

SVM scores with C=0.001 and Kernel=poly:

precision recall f1-score support

0 0.73 1.00 0.84 192

1 1.00 0.66 0.79 208

accuracy 0.82 400

macro avg 0.87 0.83 0.82 400

weighted avg 0.87 0.82 0.82 400

SVM scores with C=0.01 and Kernel=poly:

precision recall f1-score support

0 0.92 1.00 0.96 192

1 1.00 0.92 0.96 208

accuracy 0.96 400

macro avg 0.96 0.96 0.96 400

weighted avg 0.96 0.96 0.96 400

SVM scores with C=0.1 and Kernel=poly:

precision recall f1-score support

0 0.96 1.00 0.98 192

1 1.00 0.96 0.98 208

accuracy 0.98 400

macro avg 0.98 0.98 0.98 400

weighted avg 0.98 0.98 0.98 400

SVM scores with C=0.001 and Kernel=rbf:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.01 and Kernel=rbf:

precision recall f1-score support

0 0.93 0.94 0.94 192

1 0.95 0.94 0.94 208

accuracy 0.94 400

macro avg 0.94 0.94 0.94 400

weighted avg 0.94 0.94 0.94 400

SVM scores with C=0.1 and Kernel=rbf:

precision recall f1-score support

0 0.95 0.99 0.97 192

1 0.99 0.95 0.97 208

accuracy 0.97 400

macro avg 0.97 0.97 0.97 400

weighted avg 0.97 0.97 0.97 400

SVM scores with C=0.001 and Kernel=sigmoid:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.01 and Kernel=sigmoid:

precision recall f1-score support

0 0.63 0.72 0.67 192

1 0.70 0.61 0.65 208

accuracy 0.66 400

macro avg 0.67 0.66 0.66 400

weighted avg 0.67 0.66 0.66 400

SVM scores with C=0.1 and Kernel=sigmoid:

precision recall f1-score support

0 0.51 0.49 0.50 192

1 0.55 0.57 0.56 208

accuracy 0.53 400

macro avg 0.53 0.53 0.53 400

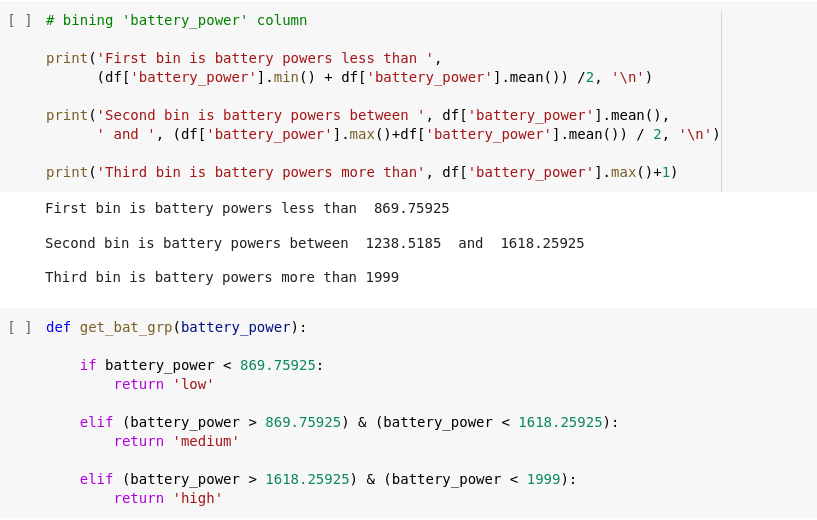
weighted avg 0.53 0.53 0.53 400

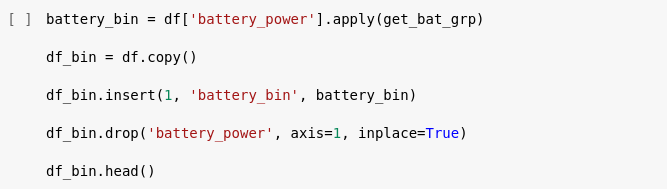
# **8)**

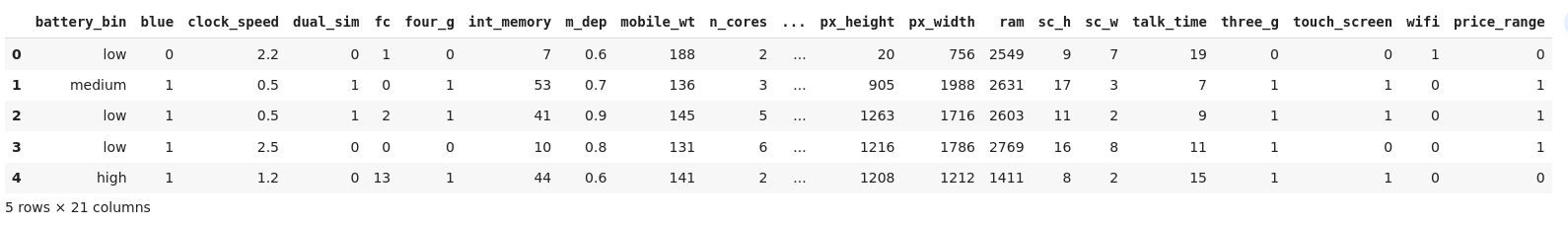
در مدل SVM پارامتر c کنترل کننده soft یا hard بودن margin است اگر مقدار خیلی بزرگی به آن بدهیم (برای مثال 1e+10) مدل دارای hard margin است و برعکس. همان طور که در نتایج بالا مشاهده می‌شود برای SVM با کرنل چند جمله‌ای c=0.001 دقت برابر با ۸۲٪ بوده (soft margin) و برای SVM با کرنل چند جمله‌ای c=0.1 دقت برابر با ۹۸٪ بوده (hard margin).

# ۹ (آ)

در این قسمت بر روی ویژگی battery\_power که دارای مقادیر پیوسته، binning انجام می‌دهیم و آن را به سه دسته low medium و high تبدیل می‌کنیم (تعداد هرکدام از این دسته‌ها متفاوت است)

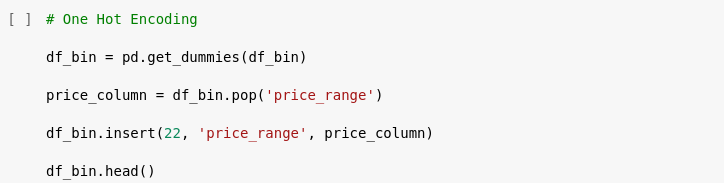


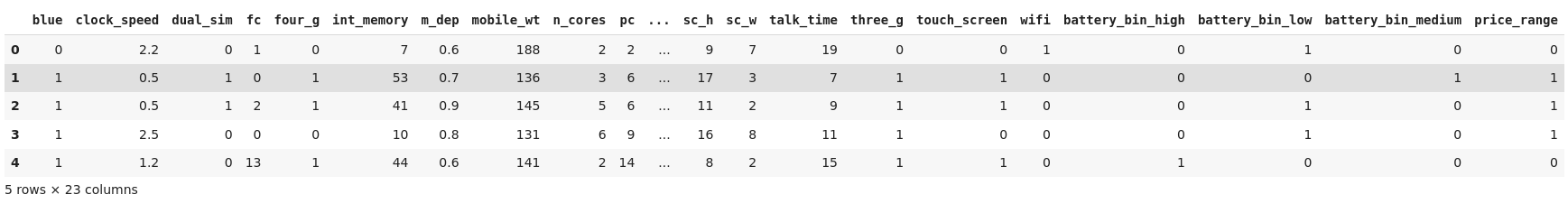


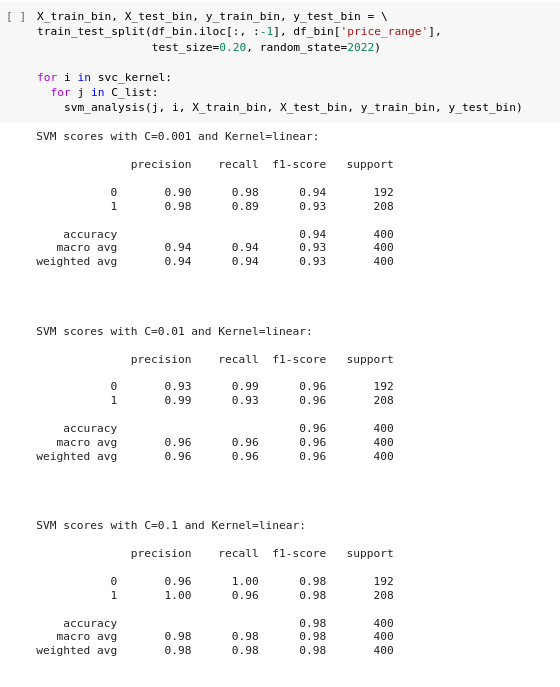


# ۹ (ب) و ۱۰

بر روی ویژگی‌های کیفی که تنها یک عدد از آن‌ها وجود دارد، OneHotEncoding را اعمال می‌کنیم و مدل‌های SVM که قبل تر گزارش شد را بر روی این دیتاست تغییر یافته هم اعمال می‌کنیم. همچنین معیار‌های خواسته شده را برای آن‌ها نمایش می‌دهیم.







SVM scores with C=0.001 and Kernel=linear:

precision recall f1-score support

0 0.90 0.98 0.94 192

1 0.98 0.89 0.93 208

accuracy 0.94 400

macro avg 0.94 0.94 0.93 400

weighted avg 0.94 0.94 0.93 400

SVM scores with C=0.01 and Kernel=linear:

precision recall f1-score support

0 0.93 0.99 0.96 192

1 0.99 0.93 0.96 208

accuracy 0.96 400

macro avg 0.96 0.96 0.96 400

weighted avg 0.96 0.96 0.96 400

SVM scores with C=0.1 and Kernel=linear:

precision recall f1-score support

0 0.96 1.00 0.98 192

1 1.00 0.96 0.98 208

accuracy 0.98 400

macro avg 0.98 0.98 0.98 400

weighted avg 0.98 0.98 0.98 400

SVM scores with C=0.001 and Kernel=poly:

precision recall f1-score support

0 0.73 1.00 0.84 192

1 1.00 0.66 0.79 208

accuracy 0.82 400

macro avg 0.87 0.83 0.82 400

weighted avg 0.87 0.82 0.82 400

SVM scores with C=0.01 and Kernel=poly:

precision recall f1-score support

0 0.86 0.99 0.92 192

1 0.99 0.85 0.91 208

accuracy 0.92 400

macro avg 0.93 0.92 0.92 400

weighted avg 0.93 0.92 0.92 400

SVM scores with C=0.1 and Kernel=poly:

precision recall f1-score support

0 0.88 0.98 0.93 192

1 0.98 0.88 0.93 208

accuracy 0.93 400

macro avg 0.93 0.93 0.93 400

weighted avg 0.94 0.93 0.93 400

SVM scores with C=0.001 and Kernel=rbf:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.01 and Kernel=rbf:

precision recall f1-score support

0 0.92 0.92 0.92 192

1 0.93 0.93 0.93 208

accuracy 0.93 400

macro avg 0.92 0.92 0.92 400

weighted avg 0.93 0.93 0.93 400

SVM scores with C=0.1 and Kernel=rbf:

precision recall f1-score support

0 0.93 0.95 0.94 192

1 0.96 0.94 0.95 208

accuracy 0.94 400

macro avg 0.94 0.95 0.94 400

weighted avg 0.95 0.94 0.95 400

SVM scores with C=0.001 and Kernel=sigmoid:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.01 and Kernel=sigmoid:

precision recall f1-score support

0 0.86 0.53 0.65 192

1 0.68 0.92 0.78 208

accuracy 0.73 400

macro avg 0.77 0.72 0.72 400

weighted avg 0.77 0.73 0.72 400

SVM scores with C=0.1 and Kernel=sigmoid:

precision recall f1-score support

0 0.53 0.48 0.51 192

1 0.56 0.61 0.59 208

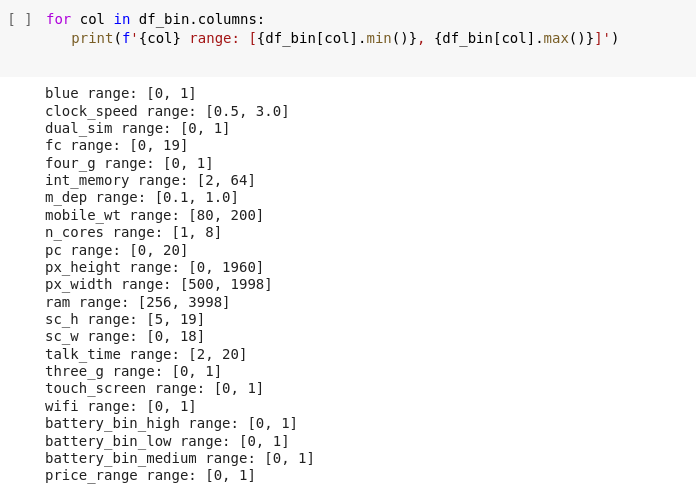
accuracy 0.55 400

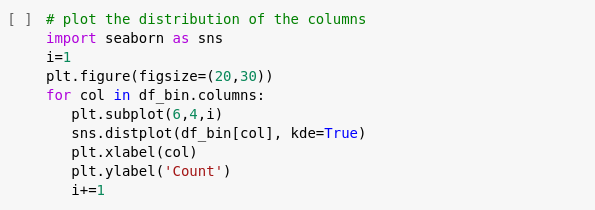
macro avg 0.55 0.55 0.55 400

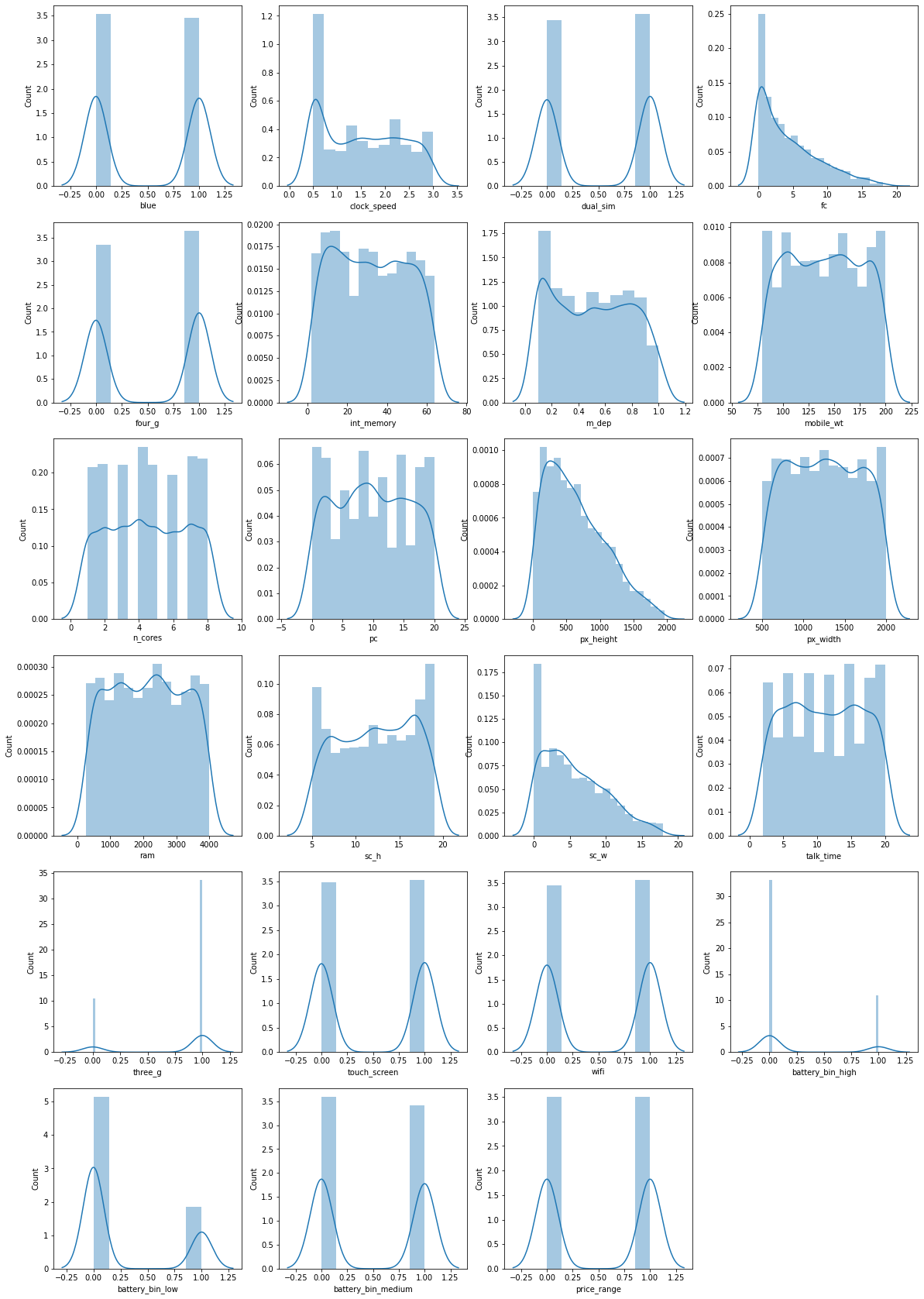
weighted avg 0.55 0.55 0.55 400

# ۹ (ج)

تبدیل لگاریتمی، در مواردی که دیتاست skew دارد کاربردی است و باعث می‌شود توزیع دادگان نرمال تر شود ابتدا بازه تغییرات ستون‌ها را در پایین چاپ می‌کنیم، و سپس توزیع احتمالی هر ستون را رسم می‌نماییم.

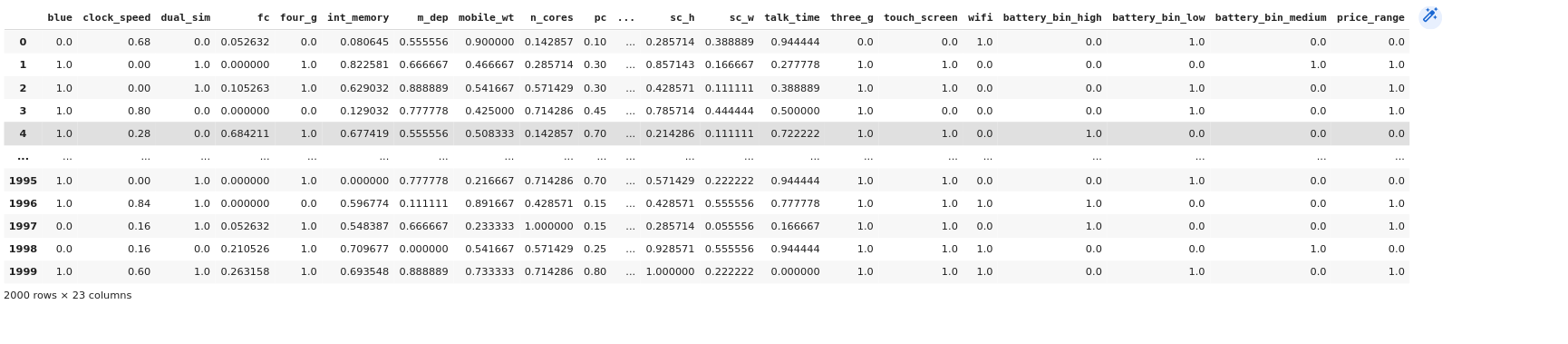


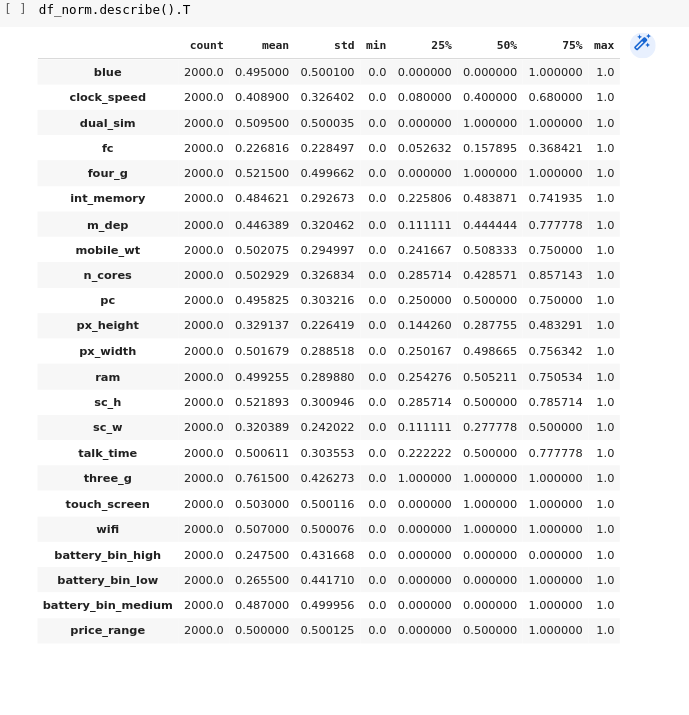




مشاهده می‌شود که ستون‌های fc، px\_height و sc\_w هر سه دارای skew می‌باشند. از آن جا که مقدار کمینه این ستون‌ها عدد صفر استُ اعمال تبدیل لگاریتمی در این مسئله امکان پذیر نمی‌باشد. با توجه به بازه‌های بدست آمده در تصویر قبل می‌توان دید مقادیر آن‌ها تفاوت بسیار زیادی از مرتبه 10**3** دارند و می‌توان برای رفع این مشکل از تبدیل استاندارد (max-min) استفاده کرد تا تمام دیتا در بازه صفر تا یک خلاصه شوند.



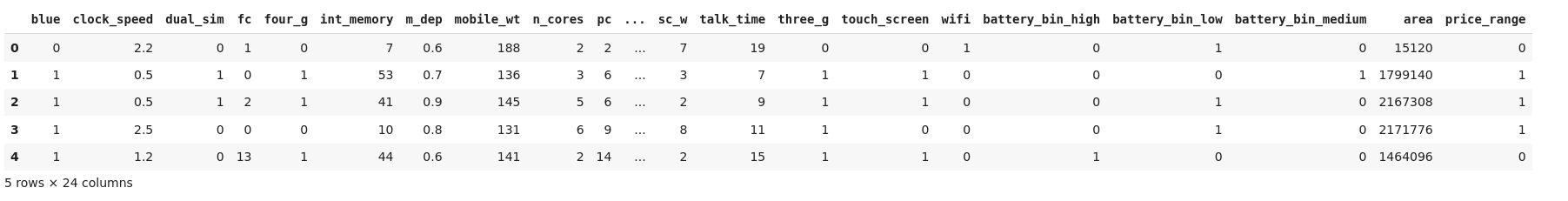


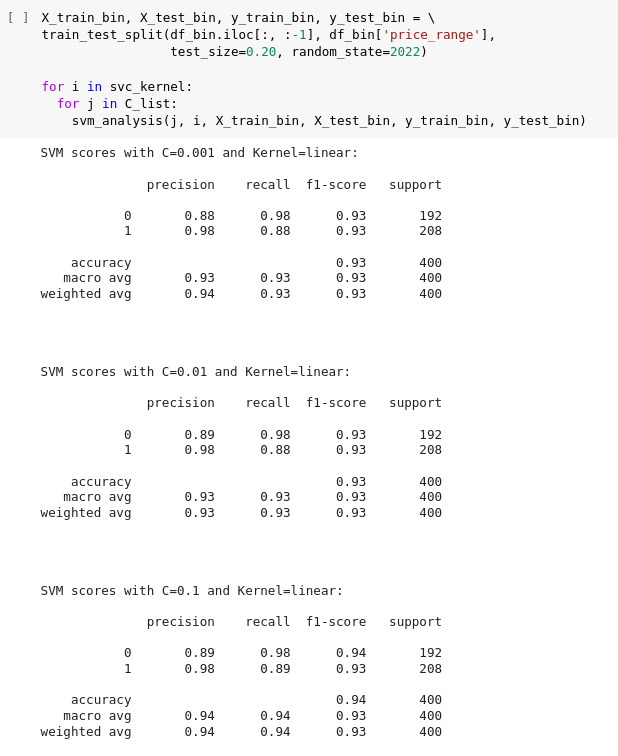


۹ (د) و ۱۰

ویژگی مساحت گوشی را با استفاده از طول و عرض آن بدست می‌آوریم. آن را به دیتا ست اضافه می‌کنیم و مدل‌های SVM را بر روی آن‌ها اعمال می‌کنیم







SVM scores with C=0.001 and Kernel=linear:

precision recall f1-score support

0 0.88 0.98 0.93 192

1 0.98 0.88 0.93 208

accuracy 0.93 400

macro avg 0.93 0.93 0.93 400

weighted avg 0.94 0.93 0.93 400

SVM scores with C=0.01 and Kernel=linear:

precision recall f1-score support

0 0.89 0.98 0.93 192

1 0.98 0.88 0.93 208

accuracy 0.93 400

macro avg 0.93 0.93 0.93 400

weighted avg 0.93 0.93 0.93 400

SVM scores with C=0.1 and Kernel=linear:

precision recall f1-score support

0 0.89 0.98 0.94 192

1 0.98 0.89 0.93 208

accuracy 0.94 400

macro avg 0.94 0.94 0.93 400

weighted avg 0.94 0.94 0.93 400

SVM scores with C=0.001 and Kernel=poly:

precision recall f1-score support

0 0.49 0.95 0.64 192

1 0.62 0.07 0.13 208

accuracy 0.49 400

macro avg 0.56 0.51 0.39 400

weighted avg 0.56 0.49 0.38 400

SVM scores with C=0.01 and Kernel=poly:

precision recall f1-score support

0 0.49 0.95 0.65 192

1 0.66 0.09 0.16 208

accuracy 0.50 400

macro avg 0.57 0.52 0.40 400

weighted avg 0.58 0.50 0.39 400

SVM scores with C=0.1 and Kernel=poly:

precision recall f1-score support

0 0.49 0.95 0.65 192

1 0.66 0.09 0.16 208

accuracy 0.50 400

macro avg 0.57 0.52 0.40 400

weighted avg 0.58 0.50 0.39 400

SVM scores with C=0.001 and Kernel=rbf:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.01 and Kernel=rbf:

precision recall f1-score support

0 0.50 0.89 0.64 192

1 0.63 0.18 0.28 208

accuracy 0.52 400

macro avg 0.56 0.53 0.46 400

weighted avg 0.57 0.52 0.45 400

SVM scores with C=0.1 and Kernel=rbf:

precision recall f1-score support

0 0.49 0.77 0.60 192

1 0.56 0.27 0.37 208

accuracy 0.51 400

macro avg 0.53 0.52 0.49 400

weighted avg 0.53 0.51 0.48 400

SVM scores with C=0.001 and Kernel=sigmoid:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.01 and Kernel=sigmoid:

precision recall f1-score support

0 0.48 1.00 0.65 192

1 0.00 0.00 0.00 208

accuracy 0.48 400

macro avg 0.24 0.50 0.32 400

weighted avg 0.23 0.48 0.31 400

SVM scores with C=0.1 and Kernel=sigmoid:

precision recall f1-score support

0 0.45 0.52 0.48 192

1 0.49 0.42 0.45 208

accuracy 0.47 400

macro avg 0.47 0.47 0.47 400

weighted avg 0.47 0.47 0.47 400

# 11)

درخت های تصمیم گیری از قدرتمند ترین الگوریتم ها هستند که به عنوان زیر مجموعه ای از الگوریتم های تحت نظارت(supervised algorithms) در نظر گرفته می شوند. الگوریتم های مختلفی برای ساخت درخت تصمیم وجود دارند. الگوریتم درخت تصمیم به گونه ای عمل می کند که سعی دارد گوناگونی و یا تنوع را در گره ها به حداقل ممکن برساند. این عدم یکنواختی در گره ها با استفاده از معیارهای عدم خلوص قابل اندازه گیری است که مهمترین و پرکاربرد ترین آن شاخص جینی می باشد. اغلب تفاوت انواع درخت های تصمیم در همین معیار اندازه گیری عدم خلوص، شیوه شاخه بندی و هرس کردن گره های درخت می باشد.

**الگوریتم CART**: برای برقراری درختهای رگرسیون و دسته بندی از این الگوریتم استفاده میشود. در سال 1984توسط Breiman و همکارانش ارائه شده است. یکی از محبوب‌ترین و در عین حال ساده‌ترین الگوریتم‌های درخت‌های تصمیم است که کاربردهای زیادی در طبقه بندی و رگرسیون دارد و بر اساس درخت های دودویی (باینری) بنا نهاده شده است. الگوریتم CART متغیرهای ورودی را برای یافتن بهترین تجزیه می آزماید تا شاخص ناخالصی حاصل از تجزیه کمترین مقدار باشد. در تجزیه دو زیر گروه تعیین می شود و هر کدام در مرحله بعد به دو زیر گروه دیگر تقسیم خواهند شد و این روند ادامه می یابد تا زمانی که یکی از معیار های توقف برآورده شود. درخت CART بازگشتی دو دویی است، که گره های والدین را دقیقا به دو گروه فرزند منشعب می کند و به طور بازگشتی منشعب کردن را تا زمانی که انشعاب دیگری نتواند ساخته شود ادامه می دهد.

**الگوریتم ID3**: یکی از الگوریتم‌­های بسیار ساده درخت تصمیم که در سال 1986 توسط Quinlan مطرح شده است. اطلاعات به دست آمده به عنوان معیار تفکیک به کار می­‌رود. این الگوریتم هیچ فرایند هرس کردن را به کار نمی­‌برد و مقادیر اسمی و مفقوده را مورد توجه قرار نمی­‌دهد. الگوریتم ID3 وظیفه پیدا کردنِ ویژگی‌هایی دارای اطلاعات زیادتر ( Gainپبیشتر) را دارد و آن‌ها را در سطوحِ بالاتری از درخت قرار می‌دهد. هر بار که یک ویژگی در سطحی از درخت انتخاب شد، زیر درخت‌های آن نیز دقیقا به همان صورت (ویژگی‌هایی با اطلاعات بالا) انتخاب می‌شوند و در سطوح و گره‌های بعدی قرار می‌گیرند. البته وقتی یک گره از درخت انتخاب شد، برای ساختِ زیر درخت‌های دیگر، مجموعه داده‌ها بر اساس مقدارِ گره‌ی انتخاب شده در شاخه‌های بالاتر، کوچکتر می‌شوند و هر چه در درخت پایین‌تر می‌رویم (به برگ‌ها نزدیک‌تر می‌شویم)، مجموعه داده‌ها برای محاسبه‌ی مقدار اطلاعات کمتر می‌شوند.

**الگوریتم C4.5:** این الگوریتم درخت تصمیم، تکامل یافته ID3 است که در سال 1993 توسط Quinlan  مطرح شده است. ایشان بعد از اینکه به نقاط ضعفِ این الگوریتم پی‌برد، در مدتِ کوتاهی الگوریتمِ بعدی خود یعنی C4.5 را طراحی کرد. از نقاطِ ضعف الگوریتم ID3 که در C4.5 رفع شده است می‌توان به موارد زیر اشاره کرد:

۱. الگوریتم C4.5 می‌تواند مقادیر گسسته یا پیوسته را در ویژگی‌ها درک کند. ولی الگوریتمِ ID3 اولیه نمی‌تواند تفاوتِ مقادیرِ عددیِ پیوسته را درک کند.

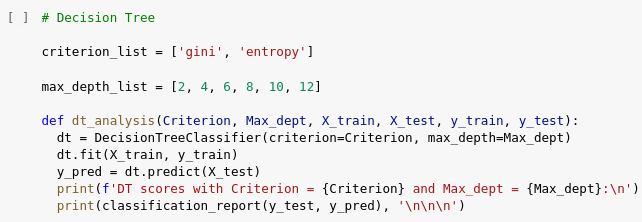
۲. الگوریتمِ C4.5 قادر است تا مقادیری که موجود نیستند را هم تحمل کند.

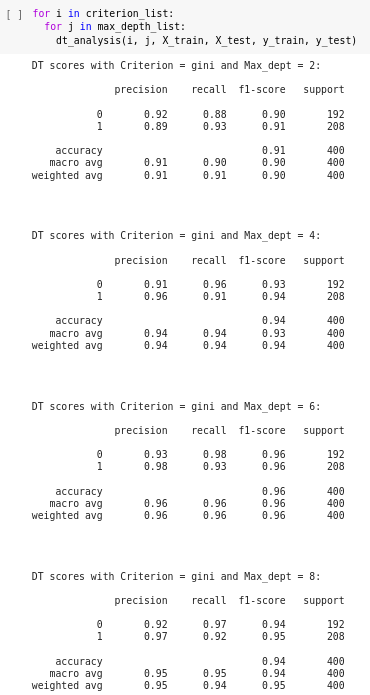
3. الگوریتم‌هایی مانند ID3 به خاطر اینکه سعی دارند تا حد امکان شاخه و برگ داشته باشند (تا به نتیجه مورد نظر برسند) با احتمال بالاتری دارای پیچیدگی در ساخت مدل می‌شوند. اما با عملیات هرس کردن درخت که در الگوریتم C4.5 انجام می‌شود، می‌توان مدل را به یک نقطه بهینه رساند که زیاد پیچیده نباشد.

4.  الگوریتم C4.5 این قابلیت را دارد که وزن‌های مختلف و غیر یکسانی را به برخی از ویژگی‌ها بدهد.

# 12) و 13)

مدل درخت تصمیم گیری را بر روی دیتاست با پارامتر‌های مختلف از جمله criterion و عمق متفاوت اعمال می‌کنیم. معیار‌های دقت را برای آن‌ها چاپ می‌کنیم.





DT scores with Criterion = gini and Max\_dept = 2:

precision recall f1-score support

0 0.92 0.88 0.90 192

1 0.89 0.93 0.91 208

accuracy 0.91 400

macro avg 0.91 0.90 0.90 400

weighted avg 0.91 0.91 0.90 400

DT scores with Criterion = gini and Max\_dept = 4:

precision recall f1-score support

0 0.91 0.96 0.93 192

1 0.96 0.91 0.94 208

accuracy 0.94 400

macro avg 0.94 0.94 0.93 400

weighted avg 0.94 0.94 0.94 400

DT scores with Criterion = gini and Max\_dept = 6:

precision recall f1-score support

0 0.93 0.98 0.96 192

1 0.98 0.93 0.96 208

accuracy 0.96 400

macro avg 0.96 0.96 0.96 400

weighted avg 0.96 0.96 0.96 400

DT scores with Criterion = gini and Max\_dept = 8:

precision recall f1-score support

0 0.92 0.97 0.94 192

1 0.97 0.92 0.95 208

accuracy 0.94 400

macro avg 0.95 0.95 0.94 400

weighted avg 0.95 0.94 0.95 400

DT scores with Criterion = gini and Max\_dept = 10:

precision recall f1-score support

0 0.93 0.98 0.95 192

1 0.98 0.93 0.96 208

accuracy 0.95 400

macro avg 0.96 0.96 0.95 400

weighted avg 0.96 0.95 0.96 400

DT scores with Criterion = gini and Max\_dept = 12:

precision recall f1-score support

0 0.94 0.97 0.96 192

1 0.98 0.94 0.96 208

accuracy 0.96 400

macro avg 0.96 0.96 0.96 400

weighted avg 0.96 0.96 0.96 400

DT scores with Criterion = entropy and Max\_dept = 2:

precision recall f1-score support

0 0.92 0.88 0.90 192

1 0.89 0.93 0.91 208

accuracy 0.91 400

macro avg 0.91 0.90 0.90 400

weighted avg 0.91 0.91 0.90 400

DT scores with Criterion = entropy and Max\_dept = 4:

precision recall f1-score support

0 0.92 0.96 0.94 192

1 0.96 0.92 0.94 208

accuracy 0.94 400

macro avg 0.94 0.94 0.94 400

weighted avg 0.94 0.94 0.94 400

DT scores with Criterion = entropy and Max\_dept = 6:

precision recall f1-score support

0 0.95 0.98 0.97 192

1 0.99 0.95 0.97 208

accuracy 0.97 400

macro avg 0.97 0.97 0.97 400

weighted avg 0.97 0.97 0.97 400

DT scores with Criterion = entropy and Max\_dept = 8:

precision recall f1-score support

0 0.95 0.96 0.96 192

1 0.96 0.96 0.96 208

accuracy 0.96 400

macro avg 0.96 0.96 0.96 400

weighted avg 0.96 0.96 0.96 400

DT scores with Criterion = entropy and Max\_dept = 10:

precision recall f1-score support

0 0.95 0.95 0.95 192

1 0.96 0.96 0.96 208

accuracy 0.95 400

macro avg 0.95 0.95 0.95 400

weighted avg 0.95 0.95 0.95 400

DT scores with Criterion = entropy and Max\_dept = 12:

precision recall f1-score support

0 0.96 0.96 0.96 192

1 0.96 0.96 0.96 208

accuracy 0.96 400

macro avg 0.96 0.96 0.96 400

weighted avg 0.96 0.96 0.96 400

مشاهد می‌شود که عمق درخت و تعداد نمونه‌های موجود در هر گره تاثیر کمی بر دقت مدل دارند. به طوری که این تاثیر با افزایش عمق ابتدا موجب افزایش دقت می‌شود.

# 14)

الگوریتم درخت تصمیم رشد درخت را بر اساس یک معیار توقف، متوقف میکند. ساده ترین معیار توقف، معیاری است که در آن همه نمونه های آموزشی در برگ متعلق به یک کلاس هستند. یک مشکل این است که ساخت درخت تصمیم تا این سطح ممکن است منجر به بیش برازش شود. چنین درختی به خوبی به نمونه های آزمایشی دیده نشده تعمیم نمی یابد. برای جلوگیری از کاهش دقتِ ناشی از بیش برازش، دسته بند از مکانیزم هرس استفاده میشود. درخت­‌های هرس شده تمایل به کوچک‌تر بودن و پیچیدگی کم­تر دارند و بنابراین به راحتی قابل فهم می­باشند. آن‌ها معمولا در طبقه‌بندی صحیح داده­‌های تست سریع­‌تر و بهتر از درخت­‌های هرس نشده عمل می­‌کنند. هرس کردن به دو روش pre-pruning و post-pruning صورت می‌‌گیرد.

پیش‌‌هرس (Pre-pruning) : در این شیوه، هرس کردن قبل از ساخت کامل درخت می‌‌باشد. به این صورت که به درختی که در حال رشد است اجازه رشد بیش از حد داده نشود. مثلا به گره تصمیمی می‌‌رسیم که در آن‌‌جا 4 آیتم مثبت و 1 آیتم منفی داریم و با توجه به آن‌‌که تمامی شروط توقف رعایت نشده است و می‌‌توان هم‌‌چنان از این گره، رشد درخت را ادامه داد ولی در این مرحله این گره تصمیم را به برگ تبدیل می‌‌نماییم و مقدار آن را بر‌‌اساس آیتم با مقادیر بیشتر که در مثال بالا آیتم‌‌های مثبت است برچسب‌‌گزاری می‌‌کنیم.

هرس بعد (Post-pruning): در این شیوه هرس کردن، درخت به صورت کامل ساخته می‌‌شود و سپس عملیات هرس کردن درخت آغاز می‌‌شود. به این‌‌صورت که از پایین درخت یا همان برگ‌‌ها به سمت ریشه حرکت می‌‌کنیم و یک‌‌سری گره‌‌های میانی را تبدیل به برگ می‌‌کنیم. این روش هرس کردن از شیوه‌‌ی قبل کمی کندتر است ولی دارای دقت بیشتری می‌‌باشد.

# 15)

بوت استرپینگ در واقع تخمین ویژگی های (مثل واریانس)یک تخمین زننده است با استفاده از اندازه گیری همین ویژگیها در یک توزیع تقریبی از کل داده های نمونه. بوت استرپینگ این امکان را برای یک نفر فراهم می سازد که تعداد زیادی نسخه ی جایگزین از یک آماره را که به طور معمول از یک نمونه محاسبه میشود را جمع آوری کند. روش bootstrap روش کارآمدی برای محاسبه میزان دقت و خطای استاندارد متغیر تخمین زده شده است. به عنوان مثال، فرض کنید که ما علاقه مند به جمع آوری اطلاعات در مورد قد افراد در جهان هستیم. به دلیل اینکه نمیتوانیم کل جمعیت را اندازه گیری کنیم، تنها یک از قسمت کوچک نمونه برداری می کنیم. از این نمونه فقط یک آماره قابل محاسبه است، مثلا یک میانگین یا یک انحراف معیار. در نتیجه نمیتوانیم متوجه شویم که آماره ها چه قدر و در چه بازه ای تغییر میکنند. اما هنگامی که از بوت استرپ استفاده کنیم ما به صورت تصادفی یک نمونه ی n تایی از N تا داده ی نمونه بر می داریم، به طوریکه هر نفر حد اکثر t بار میتواند انتخاب شود. با چندین بار انجام این کار در واقع تعداد زیادی مجموعه ی داده میسازیم که برای هرکدام میتوانیم یک آماره حساب کنیم.

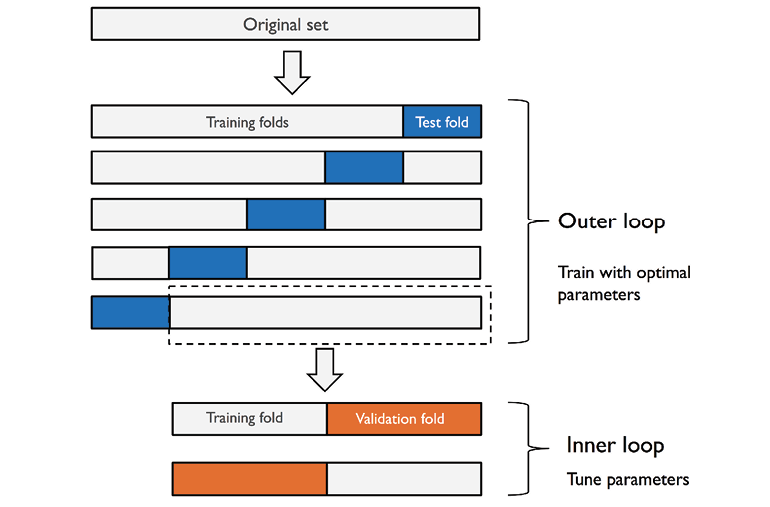
هر دو روش bootstrap و cross validation روشهای نمونه گیری مجدد هستند ولی همانظور که ذکر شد روش bootstrap برای ارزیابی میزان دقت تخمین یک پارامتر، کاربرد دارد ولی روش cross-validation، روشی برای ارزیابی عملکرد مدل و بدست آوردن تخمینی از خطای مدل است.

# 16)

اگر زمانی که داریم مدل را بر اساس یک سری از هایپرپارامترها تنظیم میکنیم، بخواهیم همزمان مدلمان را ارزیابی کنیم، باید 5x2 Cross Validation انجام دهیم. در حالت عادی وقتی این کا را انجام ندهیم ، و از همان دیتایی که برای انتخاب بهترین هایپرپارامترها استفاده کردیم، برای test کردن مدل استفاده کنیم، چون مدل در حین tuning ، داده‌ها رو تجربه کرده، test-score واقعی را نشان نخواهد داد(بهتر از چیزی که در واقع است) و داده‌ها دچار overfit میشوند، و باید زمانی که عملکرد چندین مدل را مقایسه میکنیم به آن توجه کنیم.

وقتی که از 5x2 CV استفاده می کنیم، از یک inner-cv برای انتخاب بهترین پارامتر، و از یک outer-cv برای تست کردن مدل استفاده میکنیم. بعنوان مثال (درصدها هم فرضی هستن):

اول دیتای مورد نظر را(در هر split از outer-cv) به دو قسمت 70-30 درصدی تقسیم میکنیم،بخش 30 درصدی(test set) رو برای تست نهایی(روی مدلی که tune شده) کنار می گذاریم، و بعد آن 70 درصد را به دو قسمت 80-20 تقسیم میکنیم. سپس هایپرپارامترهای مدل را (در هر split از Inner-Cv) روی این 80 درصد train می کنیم(train set) و روی بخش 20 درصدی (validation set) ، test می کنیم تا بهترین پارامتر را با توجه به عملکردشان انتخاب کنیم. بعد از اینکه که مدل با بهترین پارامترا را انتخاب کردیم، این مدل را روی 30 درصدی که در ابتدا کار کنار گذاشته بودیم test میکنیم، و این بار عملکرد واقعی‌تر و منطقی‌تری را از مدل انتظار داریم.



# 17)

از روشelbow یا آرنج برای تعیین تعداد صحیح خوشه‌ها در یک [دیتاست](https://hooshio.com/%d8%a2%d8%b4%d9%86%d8%a7%db%8c%db%8c-%d9%85%d9%82%d8%af%d9%85%d8%a7%d8%aa%db%8c-%d8%a8%d8%a7-%d8%a7%d9%86%d9%88%d8%a7%d8%b9-%d8%af%db%8c%d8%aa%d8%a7%d8%b3%d8%aa-%d8%a8%db%8c%d9%86%d8%a7%db%8c%db%8c/)استفاده می‌شود. در این روش مقادیر افزایشی k  بر روی محور افقی و مجموع خطاهایی که در هنگام استفاده از k میانگین رخ داده بر روی محور عمودی ترسیم می‌شود. هدف از استفاده از این روش یافتن kای است که برای هر خوشه واریانس را زیاد افزایش ندهد. روش آرنج درصد واریانس را به عنوان تابعی از تعداد خوشه‌ها توضیح می‌دهد: یکی باید به عنوان تعداد خوشه‌ها انتخاب شود به طوری که با اضافه کردن خوشه ای دیگر مدل‌سازی داده بهتری بدست نیاید. اگر یک ترسیم (plot) درصد واریانس را تشریح کند طوریکه مخالف تعداد خوشه‌ها باشد اولین خوشه‌ها اطلاعات زیادی (توضیح بسیاری از واریانس) را اضافه می‌کنند، اما در بعضی نقطه‌ها حاشیه سود کاهش خواهد یافت و یک زاویه در نمودار به وجود می‌آورد. تعداد خوشه‌ها در این نقطه انتخاب شده‌اند یعنی همان معیار آرنج . این آرنج نمی‌تواند همیشه به روشنی مشخص شود.



با توجه به توضیحات داده شده میتوان با توجه به trade-off بایاس و واریانس میزان مرتبه مناسب مدل را پیدا کرد. مثلا در شکل بالا درجه حدود 3 بهترین مرتبه مدل می توان در نظر گرفت. از طرفی از آنجایی که روش elbow یک روش هیوریستیک است ممکن است در همه مسایل در عمل نتوان به خوبی این کار را انجام داد.

## دیتاست شماره ۲

# 1)

قضيه بیز بیان می‌کند که اگر A و B دو واقعه‌ی مستقل باشند، احتمال رخ دادن واقعه‌ی A اگر واقعه‌ی B رخ داده باشد، برابر است با حاصل ضرب احتمال رخ دادن واقعه‌ی B در صورت رخ دادن واقعه‌ی A، در احتمال رخ دادن واقعه‌ی A تقسیم بر احتمال رخ دادن واقعه‌ی B. به عبارتی دیگر قضیه بیز، احتمال وقوع یک رویداد را بر اساس دانش قبلی از شرایطی که ممکن است مربوط به رویداد باشد، توصیف می کند.

Gaussian Naive Bayes

برای محاسبه‌ی تابع احتمال، مانند Gaussian Bayes از توابع گوسی استفاده می‌کند با این تفاوت که در این مدل از اثر ویژگی‌ها بر یکدیگر صرف نظر شده یعنی عناصر غیر قطری ماتریس کوواریانس توابع احتمال گوسی برابر با ۰ فرض می‌شوند. استفاده در مسائل دارای مقادیر پیوسته که می‌توان توسط یک توزیع احتمال گوسی آنها را مدل کرد.

Multinomial Naive Bayes

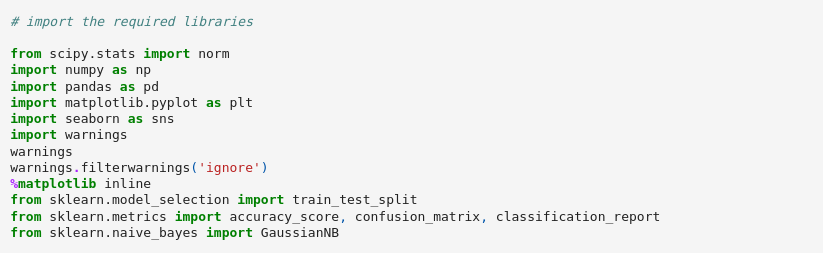
برای محاسبه‌ی تابع احتمال، از توابع multinomial استفاده می‌کند. استفاده در مسائلی که مقادیر ویژگی‌ها نشان دهنده‌ی تعداد تکرار یا به عبارتی فرکانس می‌باشد.

Bernoulli Naive Bayes

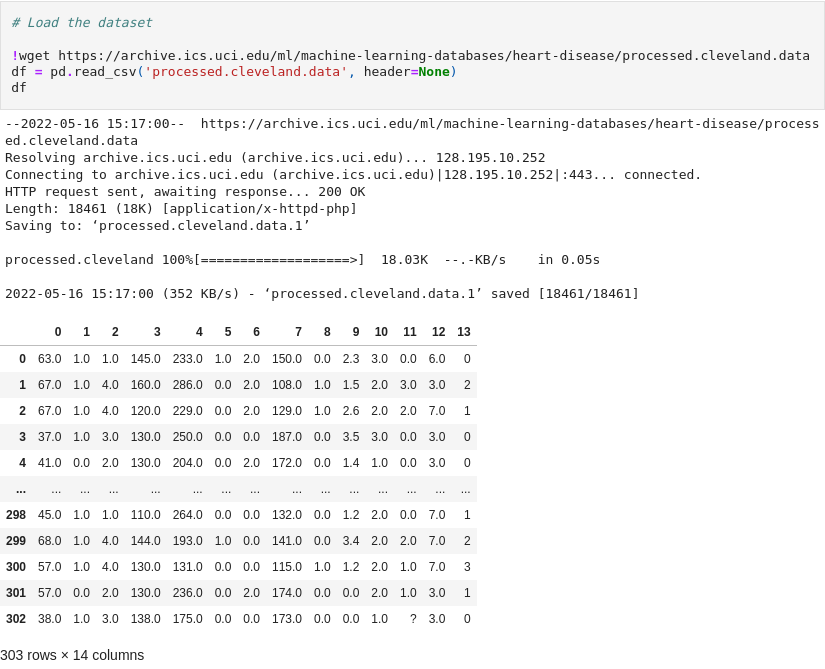
برای محاسبه‌ی تابع احتمال، از توابع bernoulli استفاده می‌کند. استفاده در مسائلی که مقادیر ویژگی‌ می‌تواند ۰ یا ۱ باشد.

# 2)

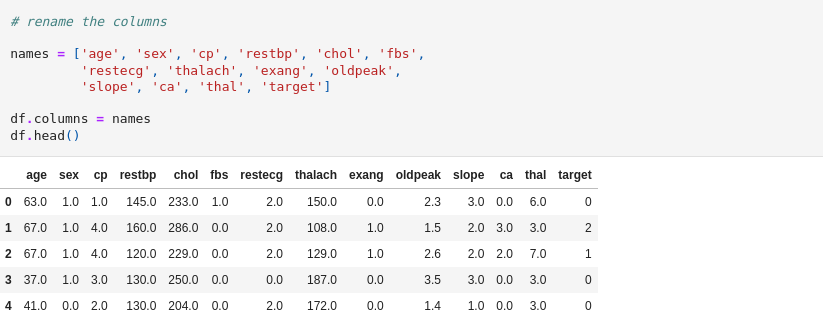
ابتدا کتابخانه‌های مورد نیاز بارگذاری میشوند:



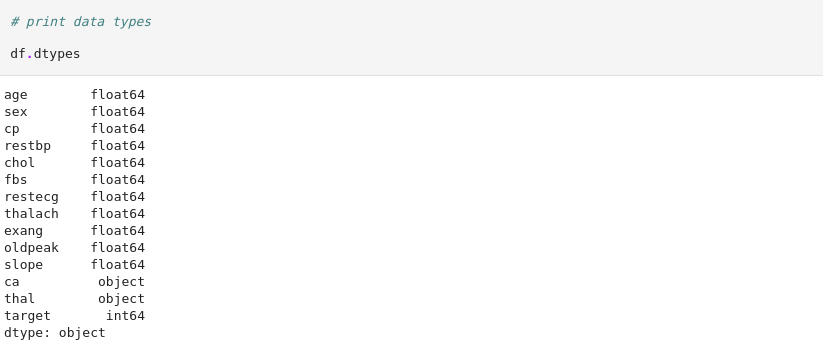
دیتاست مستقیما از لینک زیر در محیط colab بارگذاری شده و توسط ماژول pandas خوانده میشود:

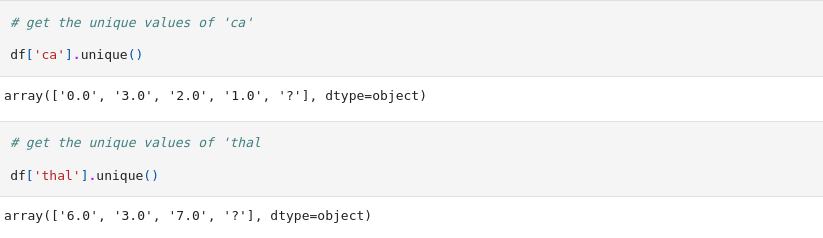


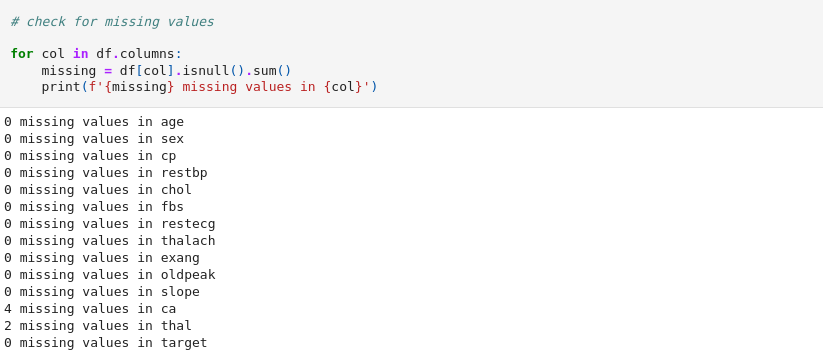
نام ستون‌ها با نام های مشخص شده جایگزین میشوند:

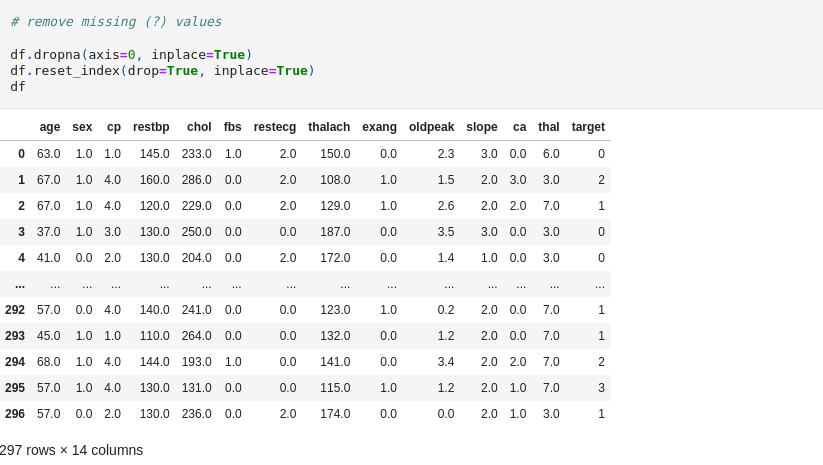


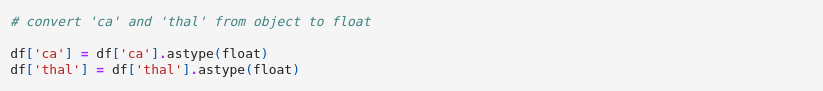
نوع دادگان هر ستون چاپ میشود و می‌بینیم در ستون‌های ca و thal مقادیر غیر عددی داریم. برای رفع این مشکل مقادیری که با ؟ مشخص شده‌اند را از دیتاست حذف کرده و نوع دادگان دو ستون ذکر شده را به float تغییر می‌دهیم.

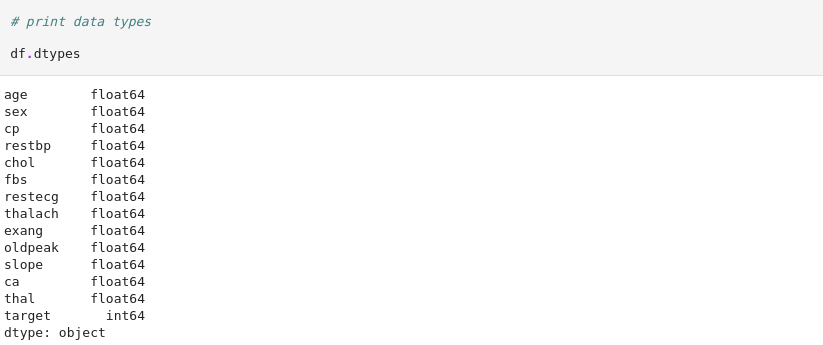




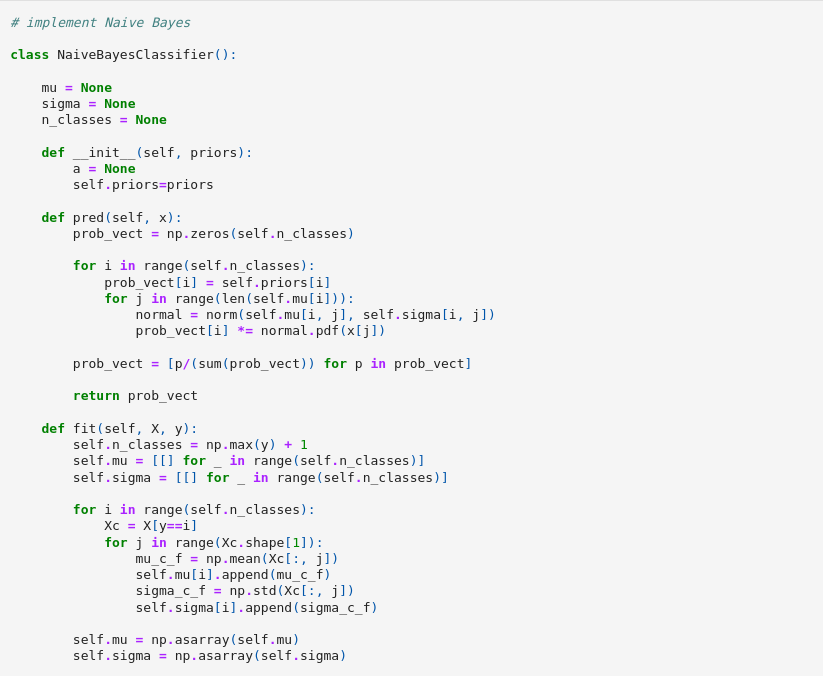








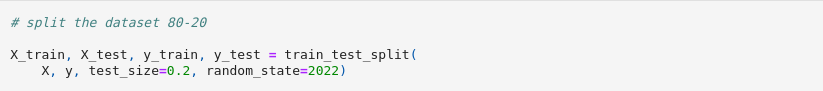
کلاس زیر برای مدل Naive Bayes classifier نوشته می‌شود:



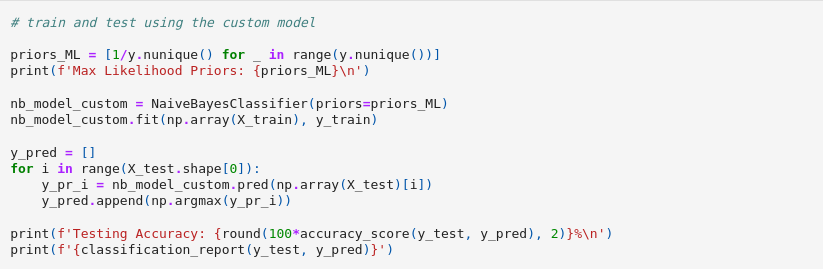
# 3)

ابتدا ماتریس ویژگی‌های ذکر شده در صورت سوال و بردار هدف تشکیل می‌شوند و سپس به صورت ۸۰-۲۰ بین train و test تقسیم میشوند:





با فرض توزیع پیشینه‌ی یکنواخت (ML) بین کلاس‌ها، مدل نوشته شده train و test میشود:



Max Likelihood Priors: [0.2, 0.2, 0.2, 0.2, 0.2]

Testing Accuracy: 43.33%

precision recall f1-score support

0 0.78 0.56 0.65 32

1 0.40 0.29 0.33 14

2 0.17 0.40 0.24 5

3 0.11 0.17 0.13 6

4 0.17 0.33 0.22 3

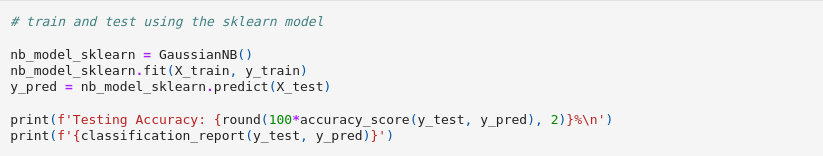
accuracy 0.43 60

macro avg 0.33 0.35 0.32 60

weighted avg 0.54 0.43 0.47 60

# 4)

مانند قسمت قبل، اینبار مدل موجود در کتابخانه‌ی sklearn استفاده میشود:



Testing Accuracy: 53.33%

precision recall f1-score support

0 0.62 0.94 0.75 32

1 0.00 0.00 0.00 14

2 0.33 0.20 0.25 5

3 0.11 0.17 0.13 6

4 0.00 0.00 0.00 3

accuracy 0.53 60

macro avg 0.21 0.26 0.23 60

weighted avg 0.37 0.53 0.43 60

# 5)

با توجه به نتایج بدست آمده در دو قسمت قبل، دقت الگوریتم نوشته شده و الگوریتم آماده در کتابخانه‌ی sklearn با هم تفاوت کمی دارند که ناشی از بهینه‌تر و دقیق‌تر بودن مدل موجود در کتابخانه‌ی sklearn می‌باشد.