

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق

درس یادگیری ماشین

استاد دکتر علیاری

سیدمحمدرضا حسینی

شماره دانشجویی: 40204584

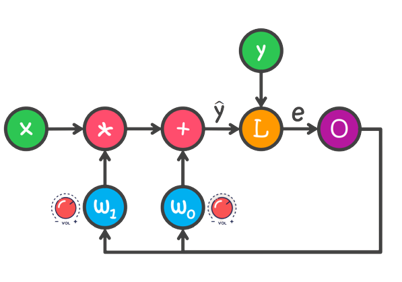
گرایش: سیستم های الکترونیک دیجیتال

مینی پروژه شماره **1**

[**Google Colab**](https://colab.research.google.com/drive/1N-tLsvbrCP4qGxUmBW8WKAe26r1T2xVo?usp=sharing)

[**Github**](https://github.com/mohammadrezahosseini99/Machine-Learning.git)

1. سوال اول
   1. بخش اول

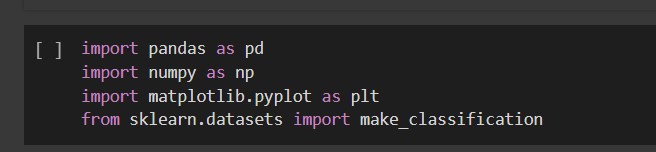


همانطور که مشاهده می شود، ویژگی ها (X) به عنوان ورودی داده می شوند. خروجی پیشبینی شده به صورت wX + w0 ساخته می شود. حال این کلاس پیشبینی شده () با تابع هزینه مناسب با کلاس واقعی (y) توسط یک تابع هزینه مناسب مقایسه می شود (e). به کمک الگوریتم های بهینه سازی (O) سعی می شود که w و w0 به گونه ای تغییر داده شوند که میزان تابع هزینه کمتر شود. این w و w0 ساخته شده مجددا به سیستم داده می شوند تا در مرحله بعدی از آنها استفاده می شود.

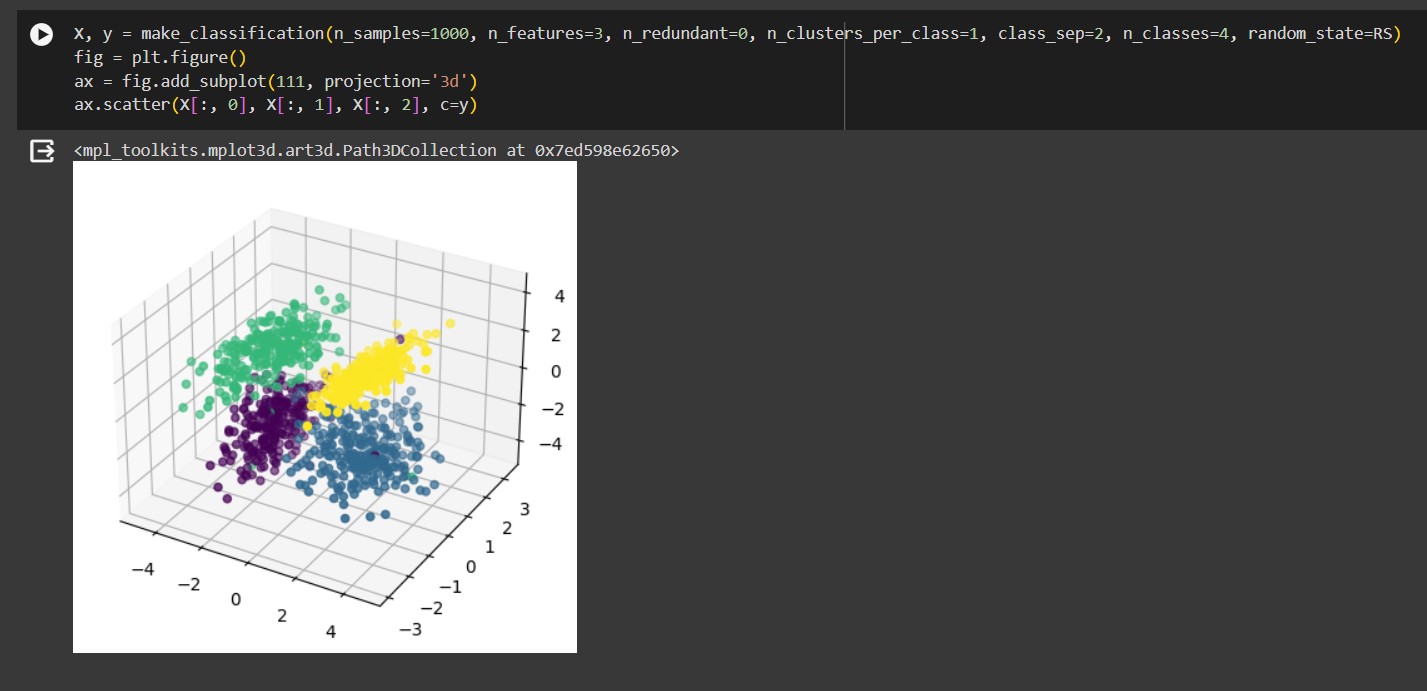
در حالت چند کلاسه، ما به جای یک طبقه بند خطی، چند طبقه بند داریم. به این شکل که یا باید تک تک کلاس ها را در مقابل بقیه کلاس ها قرار دهیم و چند طبقه بند دو کلاسه بدست آوریم یا باید کلاس ها را دو به دو با هم درنظر بگیریم و برای هر 2 کلاس متفاوت، یک طبقه بند قرار دهیم.

* 1. بخش دوم

در ابتدای کد import های مورد نیاز را انجام می دهیم.



در ادامه با توجه به خواسته سوال، دیتاست مورد نظر تولید شد. با توجه به اینکه در این سوال 3 ویژگی وجود دارد، دیتاست را در نمودار 3 بعدی نمایش دادیم.

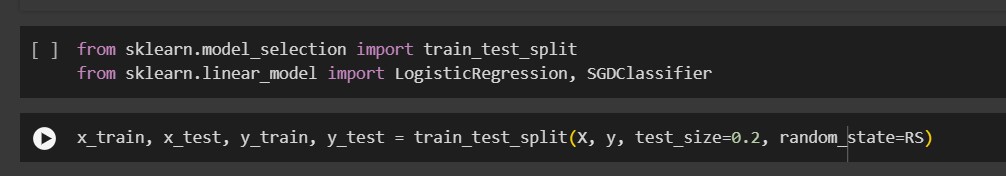


عدد class\_sep نشان دهنده میزان چالش برانگیز بودن دیتاست است. هر چه این عدد کوچکتر باشد، کلاس ها بیشتر به داخل هم فرو رفته اند و در نتیجه کار جداسازی آن ها سخت تر و چالش برانگیزتر است. به مانند تمرین حل شده در کلاس حل تمرین، در این سوال هم ما عدد class\_sep را 2 درنظر گرفتیم. با توجه به اینکه کلاس ها تقریبا به خوبی از هم جدا شده اند، این دیتاست چندان چالش برانگیر نیست. برای چالش برانگیزتر کردن آن می توان class\_sep را عددی کوچکتر مانند 0.5 قرار داد.

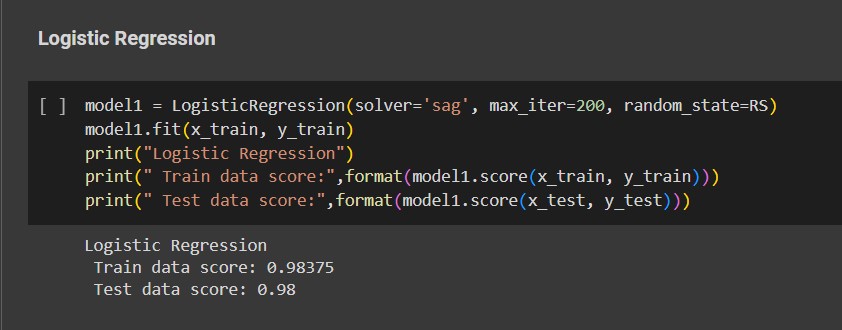
* 1. بخش سوم

در ابتدای کد import های مورد نیاز را انجام می دهیم.

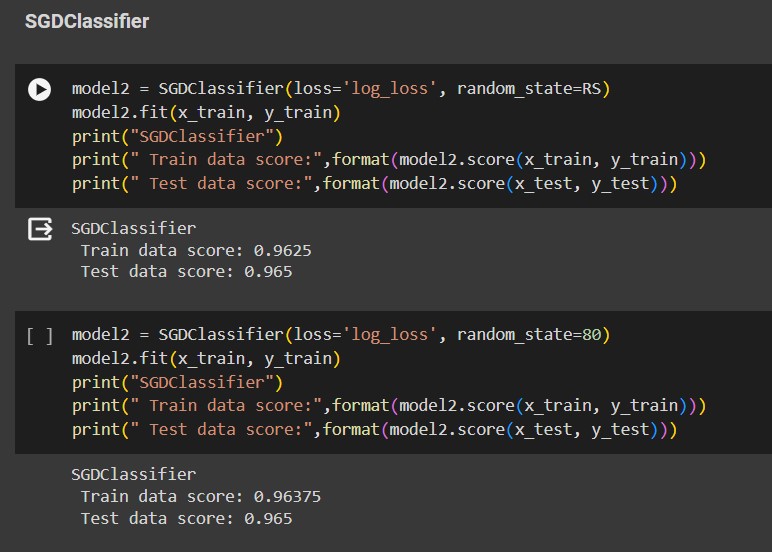
به کمک تابع train\_test\_split، دیتا را به دو قسمت train و test تقسیم می کنیم. 20 درصد برای تست و 80 درصد برای یادگیری قرار می دهیم.



در حالت اول از LogisticRegression استفاده کردیم. برای بهبود عملکرد پارامترهای مختلفی را تغییر دادیم. به طور مثال، RandomState را تغییر دادیم اما تغییر مثبتی حاصل نشد. همچنین با توجه به توضیحات [صفحه LogisticRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) در سایت scikit-learn، سایر پارامترها (مانند C و solver و ...) را تغییر دادیم اما تغییر مثبتی حاصل نشد. همچنین تعداد تکرار یا همان max\_iter را تا 20000 بالا بردیم اما بهبودی حاصل نشد. بنظر می رسد که طبقه بند همگرا شده است.



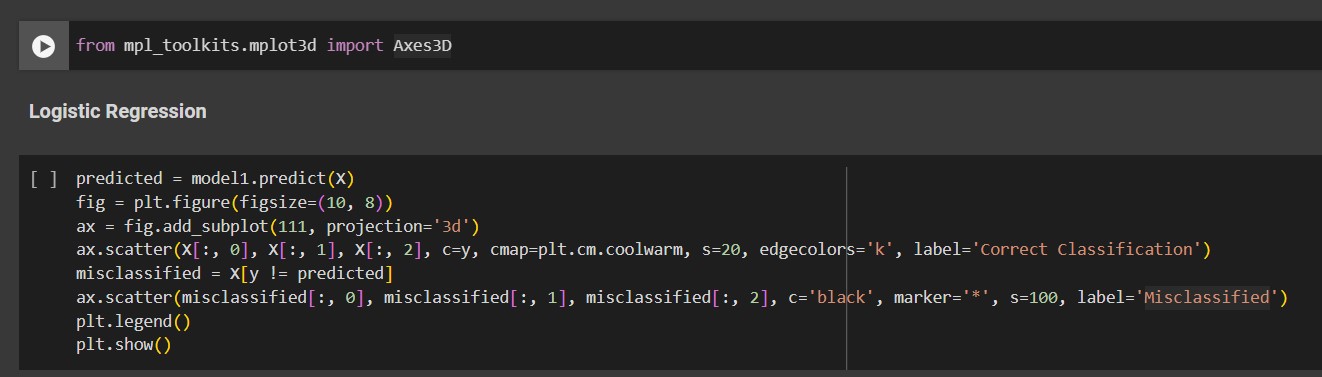
در حالت دوم از SGDClassifier استفاده کردیم. در این حالت هم مانند حالت قبل پارامترهای مختلف را با توجه به توضیحات [صفحه SGDClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html) در سایت scikit-learn تغییر دادیم و بهترین حالت را (که RandomState برابر با 80 دارد) در کد قرار دادیم.

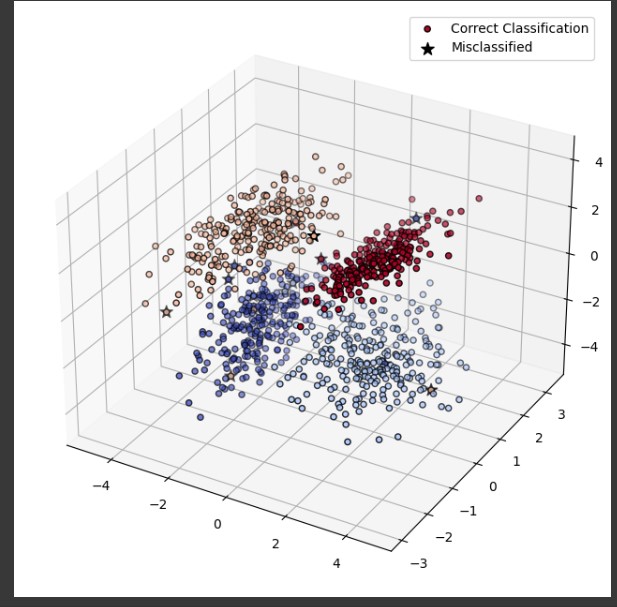


* 1. بخش چهارم

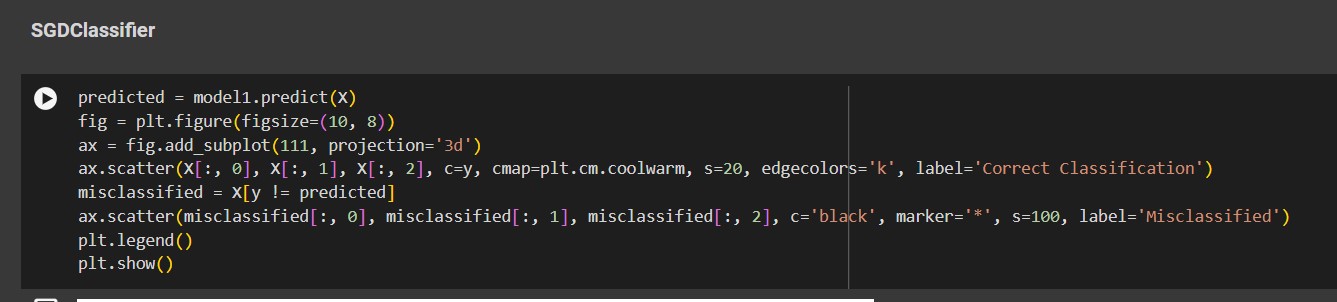
در این قسمت دیتاها در یک نمودار سه بعدی با توجه به کلاس خود رسم می شوند. آن دسته از دیتاهایی که اشتباه پیشبینی شده اند با ستاره مشخص شده اند. به دلیل اینکه تعداد ویژگی ها 3 است، مرزهای تصمیم گیری 2 بعدی و به شکل صفحه هستند که به دلیل پیچیدگی، موفق به رسم این صفحه نشدیم.

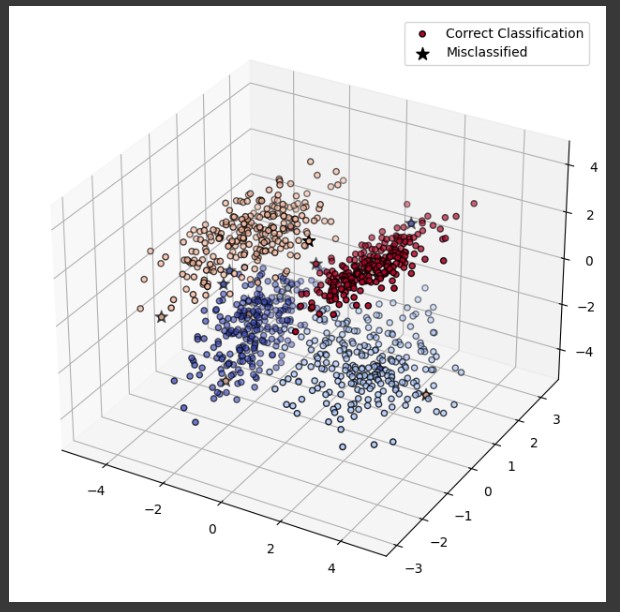
ابتدا Logistic Regression انجام شد.





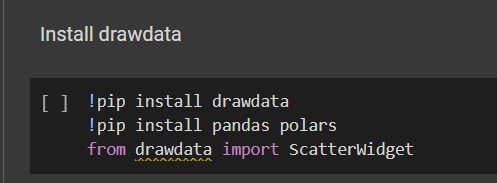
سپس SGDClassifier انجام شد.



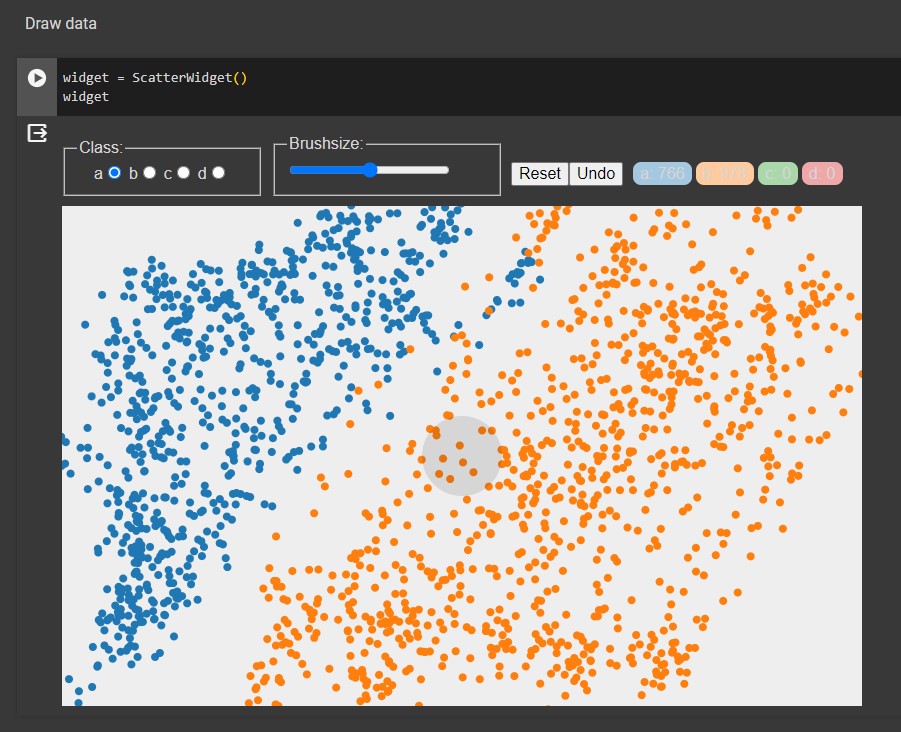


* 1. بخش پنجم

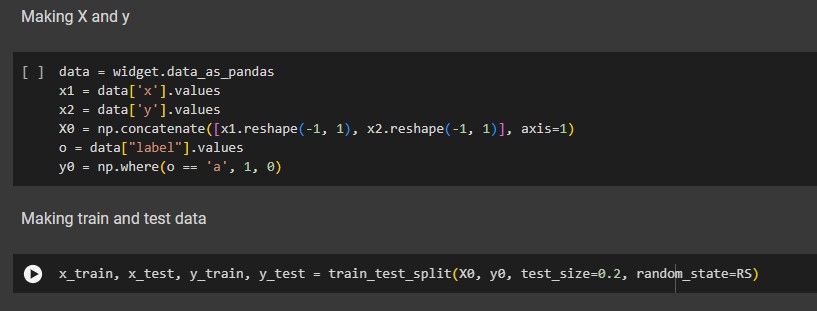
ابتدا موارد مورد نیاز را نصب می کنیم.



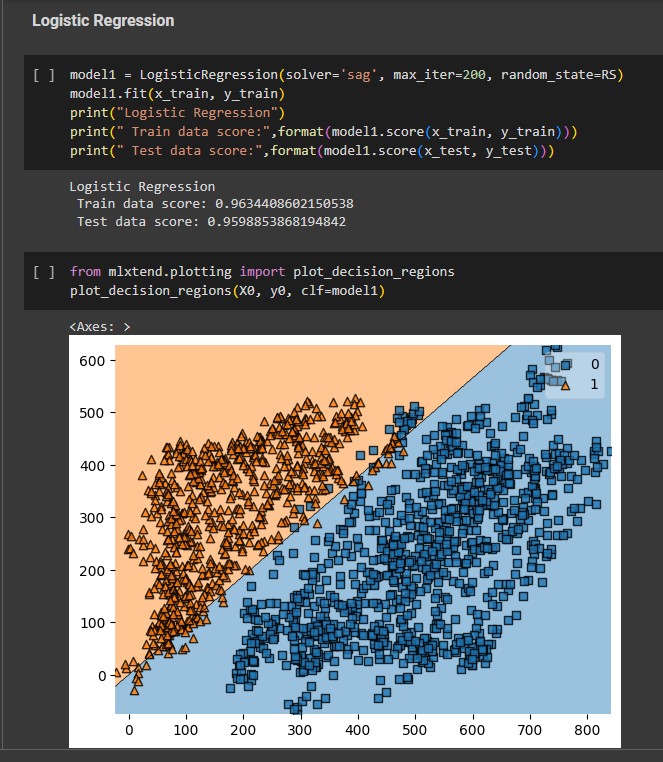
سپس یک دیتاست دو کلاسه با دو ویژگی را به کمک rawdata، رسم می کنیم.



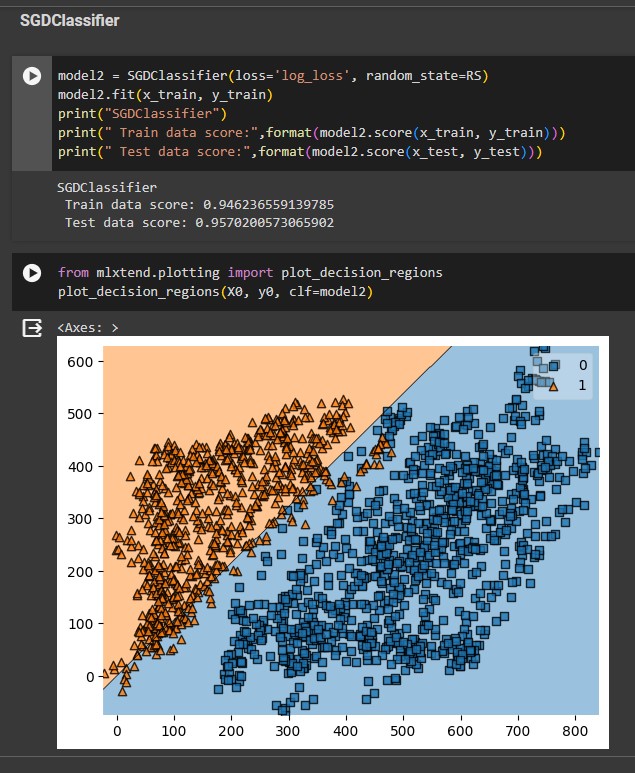
حال مقدار X و y را دریافت می کنیم و دیتا را به دو بخش آموزش و تست تقسیم می کنیم.



در مرحله اول به کمک Logistic Regression دو کلاس را از یکدیگر جدا می کنیم. Score را نمایش می دهیم و دیتاست، مرز و کلاس ها را نمایش می دهیم.



در مرحله بعد نیز تمام کارهایی که با Logistic Regression انجام دادیم را با SGDClassifier انجام می دهیم.



1. سوال دوم
   1. بخش اول

در دیتاست مربوط به حوزه CWRU Bearing، مجموعه‌ای از داده‌های ارتعاش بلبرینگ‌ها جمع‌آوری شده است، که از سه نوع بلبرینگ مختلف (بلبرینگ‌های معمولی، بلبرینگ‌های انتهای درایو و بلبرینگ‌های انتهای فن) و از حالت‌های عادی و عیب‌دار آن‌ها، نمونه‌هایی جمع‌آوری شده است.

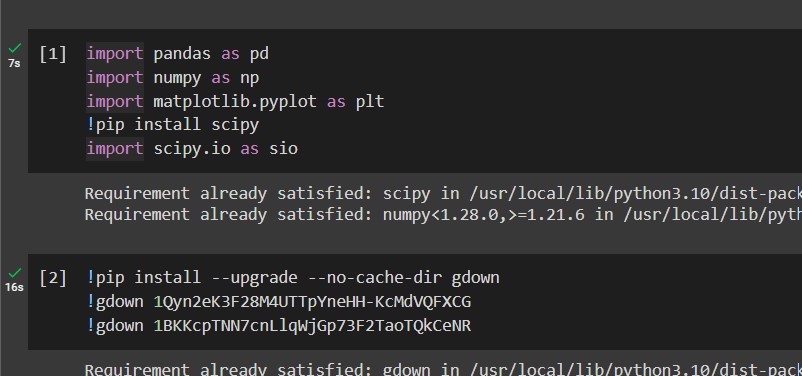
در این داده‌ها، متغیرهای مختلفی از جمله داده‌های ارتعاش انتهای فن و انتهای درایو، سرعت چرخش موتور، داده‌های شتاب‌نگاری پایه، و داده‌های سری زمانی موجود است. فرمت فایل‌ها به صورت متلب می‌باشد که این فرمت سازگاری و آسانی در تجزیه و تحلیل داده‌ها را فراهم می‌کند.

هدف اصلی این دیتاست، بهبود قابلیت اطمینان دستگاه‌های مکانیکی با شناسایی دقیق و طبقه‌بندی عیوب بلبرینگ است. با تحلیل داده‌ها و الگوهای ارتعاش مرتبط، متخصصان قادرند عیوب بلبرینگ را به دقت شناسایی و طبقه‌بندی کنند و از طریق این شناسایی، بهبود استراتژی‌های نظارت بر وضعیت، توسعه و اعتبارسنجی الگوریتم‌های تشخیص عیب، بهبود مدل‌های نگهداری پیش‌بینی و بررسی اثرات عیوب مختلف بر ویژگی‌های ارتعاش را انجام دهند.

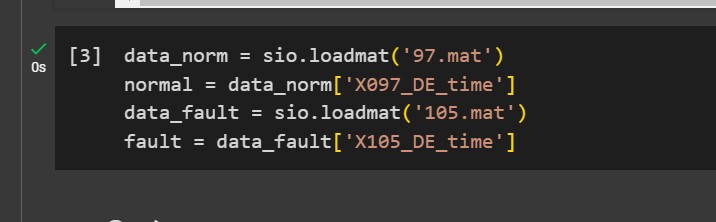
دیتاست CWRU Bearing به محققان امکان می‌دهد الگوریتم‌های تشخیصی را توسعه دهند و بهبود بخشند، مدل‌های نگهداری را بهینه‌سازی کنند، و به دقت عیوب بلبرینگ را از طریق تجزیه و تحلیل دقیق الگوهای ارتعاش و ویژگی‌های مرتبط شناسایی کنند.

با توجه به شماره دانشجویی 40204584، برای داده نرمال از Normal\_0 و برای داده خطا از IR007\_0 استفاده شده است.

در ابتدای کد، import های مورد نیاز را انجام می دهیم. همچنین به کمک دستور gdown دیتاست ها را که قبلا در Google Drive به صورت public قرار داده ایم، فرا می خوانیم.

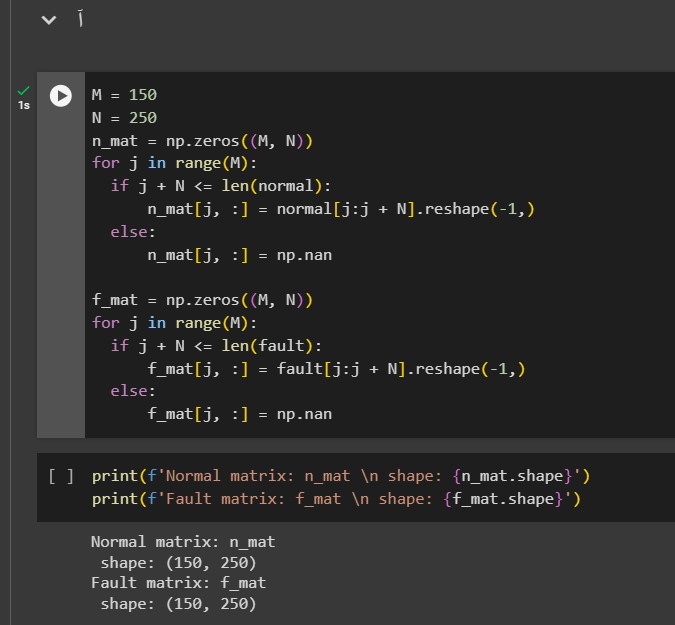


حال دیتای مورد نیاز که شامل X097\_DE\_time و X105\_DE\_time است را از دیتاست استخراج می کنیم.



* 1. بخش دوم
     + 1. قسمت آ

یک ماتریس 250×150 برای داده نرمال (n\_mat) و یک ماتریس 250×150 برای داده خطا (f\_mat) تشکیل می دهیم. (مقدار M را 150 و مقدار N را 250 در نظر گرفتیم.) به این منظور برای تشکیل هر یک از ماتریس ها یک حلقه for ایجاد کردیم که N تا N تا دیتا برمی دارد و در ماتریس قرار می دهد.



* + - 1. قسمت ب

بیشترین اهمیت استخراج ویژگی‌ها در یادگیری ماشین به این دلیل است که:

افزایش دقت مدل: با انتخاب ویژگی‌های مناسب و حذف ویژگی‌های غیرضروری، دقت مدل‌های یادگیری ماشین افزایش می‌یابد.

کاهش ابعاد داده: با استخراج ویژگی‌های مهم و کاهش ابعاد داده، فرآیند یادگیری سریع‌تر و موثرتر می‌شود و همچنین حجم داده‌های پردازشی کاهش می‌یابد که این باعث بهینه‌سازی استفاده از منابع محاسباتی می‌شود.

کاهش تأثیر نویز: با حذف داده‌های تکراری و نویزها، مدل‌های یادگیری ماشین به دقت بیشتری دست می‌یابند و توانایی تعمیم بهتری دارند.

افزایش تفهیم قابلیت‌های داده: با استخراج ویژگی‌ها، محلول‌های یادگیری ماشین قابلیت تفسیر و فهم داده‌ها را افزایش می‌دهند که این امر می‌تواند در تصمیم‌گیری‌های مربوط به کسب و کار مفید باشد.

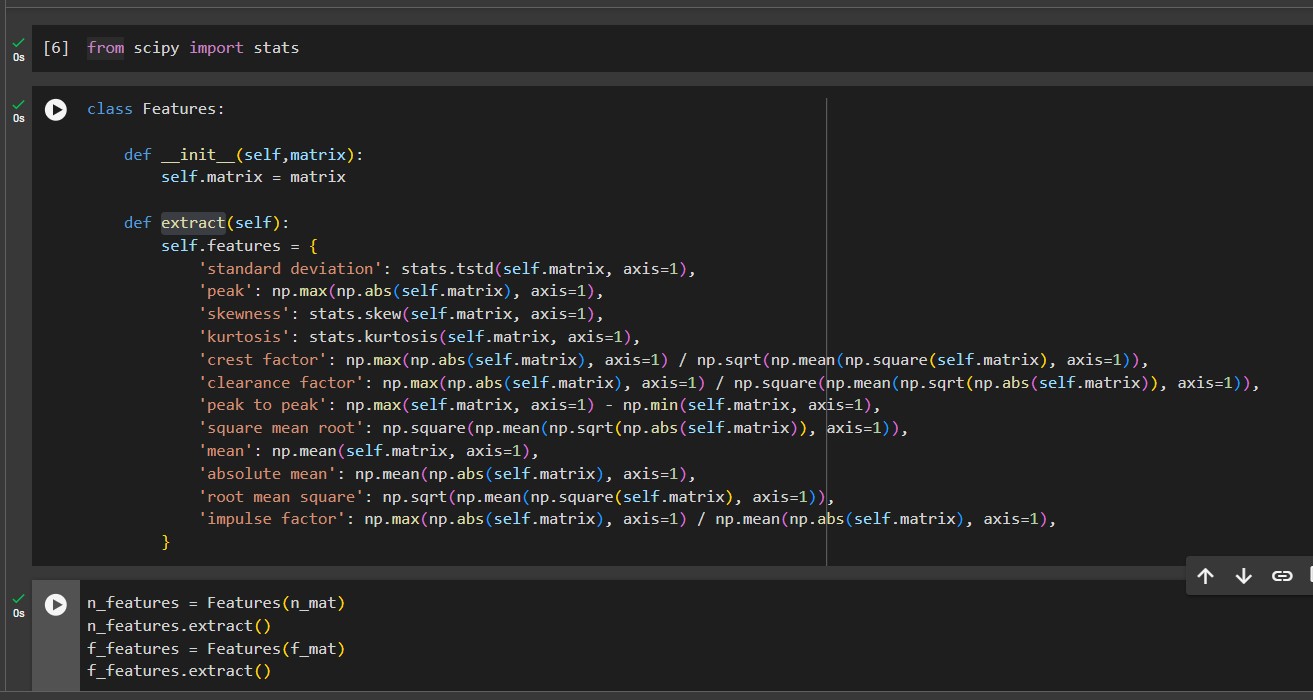
افزایش توانایی تعمیم: با استفاده از ویژگی‌های مناسب، مدل‌های یادگیری ماشین توانایی تعمیم بهتری به داده‌های جدید را دارا می‌شوند و عملکرد بهتری در مواجهه با داده‌های ناشناخته ارائه می‌دهند.

در کل، استخراج ویژگی از داده‌های خام به مدل‌های یادگیری ماشین اجازه می‌دهد تا به طور کارآمد عمل کنند و پیش‌بینی‌های دقیقی ارائه دهند.

مانند قسمت قبل ابتدا ، import مورد نیاز را انجام می دهیم. سپس یک کلاس به نام Features تعریف می کنیم. این کلاس 2 متد دارد:

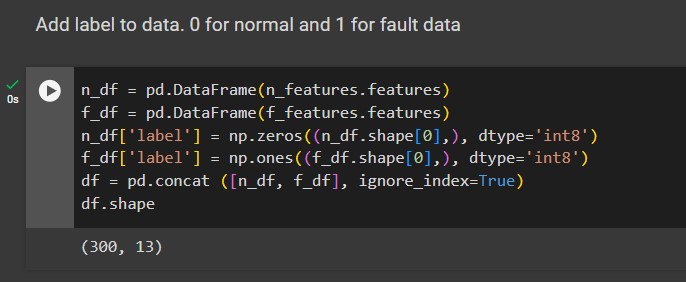
1. متد \_\_init\_\_ که برای راه اندازی و مقداردهی اولیه کلاس است و ماتریس ورودی را ذخیره می کند.
2. متد extract که ویژگی ها را از ماتریس استخراج می کند و در کلاس ذخیره می کند.

به کمک این کلاس ویژگی ها را از 2 ماتریس نرمال و خطا استخراج می کنیم.



حال که 12 ویژگی ساختیم، لازم است یک ستون به نام label جهت تشخیص کلاس دیتا اضافه کنیم. سپس این

دو دیتاست را به هم می چسبانیم تا دیتای نهایی ما بدست آید.



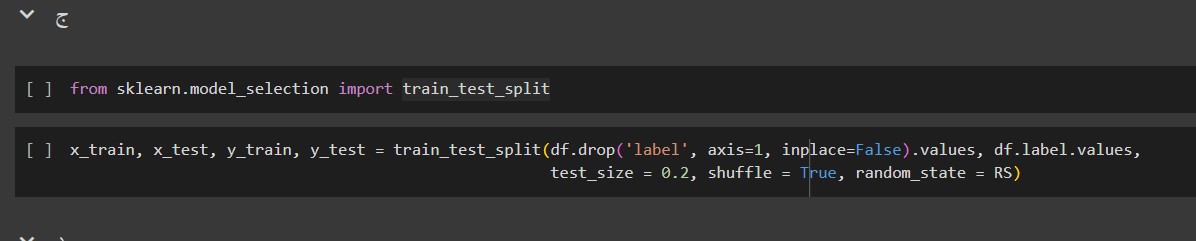
* + - 1. قسمت ج

اهمیت فرایند بر زدن:

1. جلوگیری از تعصب: بر زدن داده‌ها کمک می‌کند تا جلوی تعصب در یادگیری گرفته شود و هر دسته یا تکرار آموزش شامل نمایندگی متنوعی از داده باشد.
2. بهینه‌سازی نزول گرادیان تصادفی (SGD): بر زدن داده‌ها به خصوص هنگام استفاده از الگوریتم‌های بهینه‌سازی تصادفی مانند SGD اهمیت دارد. این الگوریتم‌ها بر روی تصادفی بودن برای فرار از کمینه‌های محلی و پیدا کردن کمینه سراسری تابع از دست رفتگی وابسته‌اند.
3. تصادفی بودن داده‌های آموزشی: برزدن داده‌ها اطمینان می‌دهد که ترتیب نمونه‌های داده بر روی فرآیند یادگیری تأثیر نگذارد و مدل به طور ناخودآگاه الگوهایی بر اساس ترتیب داده‌ها یاد نگیرد.
4. کاهش بیش‌برازش: بر زدن داده‌ها می‌تواند با واریانس در فرآیند آموزش، به کاهش بیش‌برازش کمک کند و مدل را مجبور به یادگیری نمایندگی‌های قوی‌تری از داده کند.
5. تعمیم بهتر: بر زدن به مدل کمک می‌کند تا به داده‌های ناشناخته بهتری تعمیم بدهد و از بیش‌برازش جلوگیری کند.

در نهایت، بر زدن داده‌ها یک مرحله پیش‌پردازش ضروری است که به بهبود کارایی و کارآیی مدل‌های یادگیری ماشین کمک می‌کند.

مانند قسمت های قبل در این قسمت نیز برای تقسیم دیتا از train\_test\_split استفاده شد و 20 درصد برای تست در نظر گرفته شد. با این فرق که این بار بر زدن را فعال کردیم.



* + - 1. قسمت د

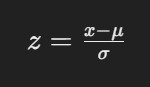
س نرمال‌سازی داده‌ها یک فرایند مهم در پیش‌پردازش داده‌های مورد استفاده در ماشین لرنینگ است. این فرآیند به معنای تغییر مقیاس داده‌ها به گونه‌ای است که مقادیر آنها در محدوده خاصی قرار گیرند، معمولاً بین ۰ و ۱ یا با میانگین صفر و واریانس یک. این کار مزایای زیادی دارد که شامل موارد زیر می‌شود:

1. جلوگیری از تأثیر مقیاس متفاوت ویژگی‌ها: ویژگی‌های با مقیاس‌های مختلف ممکن است تأثیر متفاوتی در مدل داشته باشند. با نرمال‌سازی داده‌ها، تأثیر این اختلافات مقیاس کاهش می‌یابد و مدل می‌تواند بهتر و کارآمدتر با ویژگی‌ها برخورد کند.
2. کاهش آثار نوفه و داده‌های پرت: در برخی موارد، داده‌های پرت یا نوفه می‌توانند تأثیر منفی بر عملکرد مدل داشته باشند. نرمال‌سازی داده‌ها می‌تواند به کاهش این تأثیرات کمک کند و مدل را از پایداری و قابلیت عمومی بیشتری برخوردار سازد.

برای نرمال‌سازی داده‌ها، دو روش رایج عبارتند از:

StandardScaler:

این روش داده‌ها را به گونه‌ای نرمال می‌کند که میانگین آنها صفر و واریانس آنها یک شود. فرمول استفاده شده برای نرمال‌سازی داده‌ها به صورت زیر است:



که در آن:

- x مقدار اولیه داده

- μ میانگین داده‌ها

- σ انحراف معیار داده‌ها

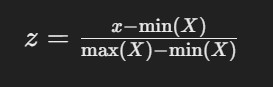
- z مقدار نرمال‌شده

است.

این روش به طور ویژه در الگوریتم‌هایی مانند SVM و نوع‌های خاصی از روش‌های خوشه‌بندی که بر اساس فاصله کار می‌کنند، مفید است. اهمیت استفاده از این روش این است که با استفاده از مقیاس یکسان برای ویژگی‌ها، مدل می‌تواند بهتر و سریع‌تر به جستجوی الگوها و ارتباط‌های مهم در داده‌ها بپردازد.

Min-Max Scaler:

این روش داده‌ها را به گونه‌ای نرمال می‌کند که مقادیر آنها بین بازه‌ای خاص (معمولاً ۰ تا ۱) قرار بگیرند. فرمول استفاده شده برای این روش به صورت زیر است:



که در آن:

- x مقدار اولیه داده

- min(X) حداقل مقدار در داده‌ها

- max(X) حداکثر مقدار در داده‌ها

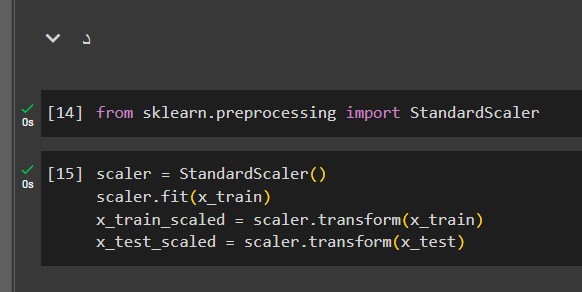
- z مقدار نرمال‌شده

است.

این روش برای الگوریتم‌هایی مانند شبکه‌های عصبی که مقادیر ورودی باید در بازه مشخصی قرار داشته باشند، مناسب است.

در مورد استفاده از داده‌های test برای نرمال‌سازی، عموماً از آنها برای محاسبه مقادیر میانگین و واریانس استفاده نمی‌شود. دلیل این امر این است که هدف از نرمال‌سازی داده‌ها ایجاد یک مقیاس یکسان برای آموزش مدل است، بنابراین اطلاعات test ممکن است نرمال شوند، اما مقادیر میانگین و واریانس بر اساس داده‌های آموزش محاسبه می‌شوند تا از انحراف مدل از داده‌های آموزش جلوگیری شود و عملکرد مدل روی داده‌های جدید بهبود یابد.

برای حل تمرین از روش StandardScaler و داده ی train استفاده شده است.

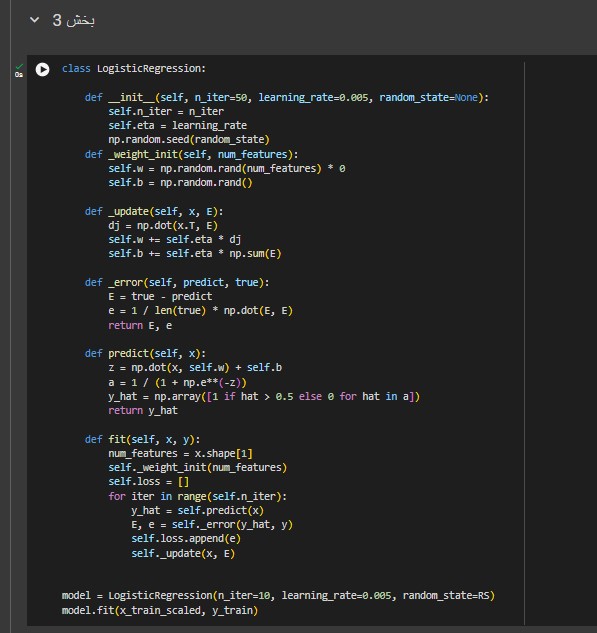


* 1. بخش سوم

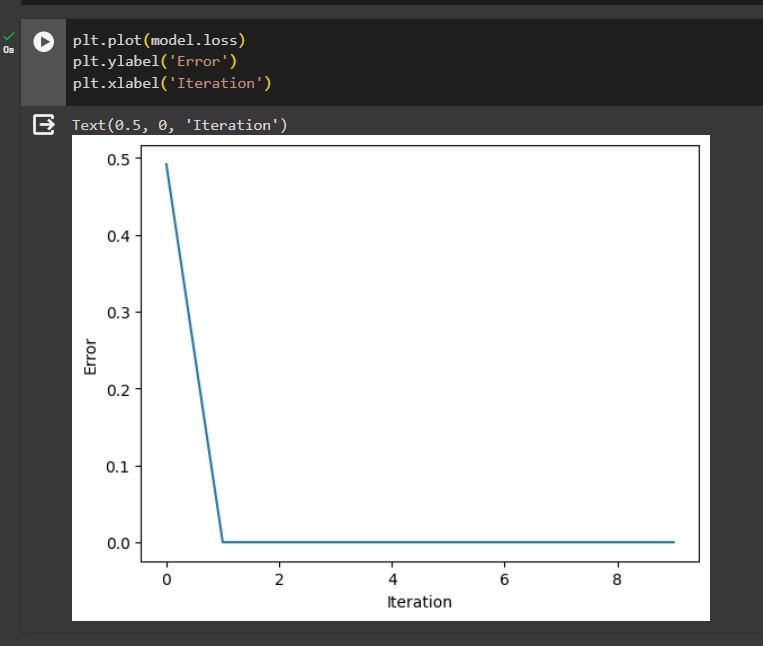
برای پیاده سازی یک مدل طبقه بند از طبقه بند LogisticRegression استفاده کردیم.

ابتدا یک کلاس به همین نام تعریف می کنیم. این کلاس شامل متد \_\_init\_\_ (مقداردهی اولیه)، \_weight\_init (مقداردهی اولیه ضرایب w یا همان وزن ها)، \_update (به روزرسانی وزن ها)، \_error (به روزرسانی خطا)، predict (پیشبینی خروجی بر حسب ورودی دلخواه) و fit (تعلیم ماشین) است.

از این کلاس برای طراحی طبقه بند دو طبقه دیتای خود استفاده کردیم. طبق تست متوجه شدیم که به دلیل راحت بودن کار تفکیک، 10 تکرار کافی است.



نمودار خطا بر حسب تکرار



از دو معیار زیر برای ارزیابی عملکرد استفاده می کنیم. Accuracy و F1-score دو معیار ارزیابی متداول در مسائل دسته‌بندی ماشینی هستند که به ترتیب میزان دقت کلی مدل و توازن بین دقت و بازخوانی را اندازه‌گیری می‌کنند.

1. Accuracy:

- دقت معیاری است که نشان می‌دهد چه تعداد از نمونه‌های دسته‌بندی شده به درستی تشخیص داده شده‌اند.

- به صورت ریاضی، دقت برابر است با تعداد نمونه‌هایی که به درستی دسته‌بندی شده‌اند تقسیم بر کل تعداد نمونه‌ها.

- این معیار مفید است زمانی که کلاس‌ها در داده‌ها متوازن هستند، به این معنا که تعداد نمونه‌ها در هر کلاس به یکدیگر نزدیک است.

2. F1-score:

- F1-score یک معیار جامع است که هم دقت (Precision) و هم بازخوانی (Recall) را در نظر می‌گیرد.

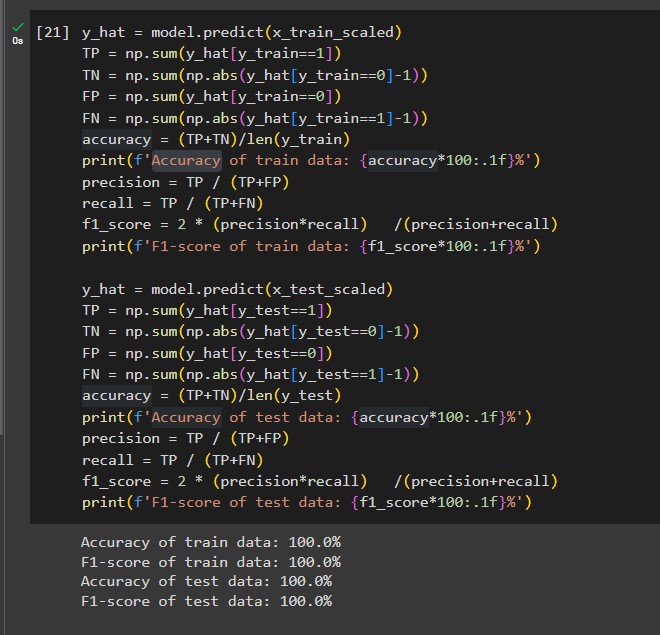
- دقت نسبت تعداد نمونه‌هایی که به درستی دسته‌بندی شده‌اند به کل نمونه‌های دسته‌بندی شده است.

- بازخوانی نسبت تعداد نمونه‌هایی که به درستی دسته‌بندی شده‌اند به کل نمونه‌های واقعی یک کلاس است.

- F1-score از این دو معیار به صورت هندسی میانگین گرفته می‌شود و به عنوان معیاری برای ارزیابی توازن بین دقت و بازخوانی مدل استفاده می‌شود.

- معمولاً زمانی که دقت و بازخوانی هر دو مهم هستند و می‌خواهیم توازن مناسبی بین آنها داشته باشیم، از F1-score استفاده می‌کنیم، به خصوص زمانی که کلاس‌ها نامتوازن هستند و تعداد نمونه‌های هر کلاس متفاوت است.

در کد زیر این دو فاکتور را برای دیتای test و train محاسبه کرده ایم:

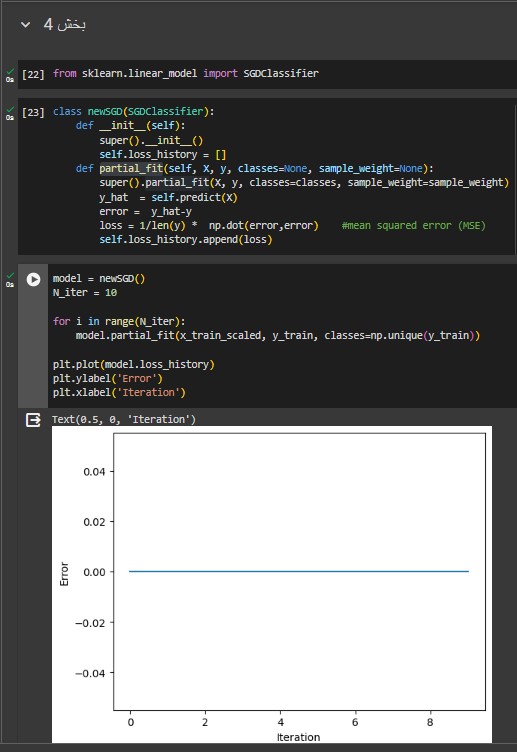


نمی توان از روی نمودار تابع اتلاف و قبل از مرحلۀ ارزیابی با قطعیت در مورد عملکرد مدل نظر داد. زیرا این نمودار حین آموزش و از طریق دیتای train ساخته می شود. ممکن است ماشین ما با این دیتا overtrain شده باشد و صرفا برای آن دیتا خیلی خوب کار کند و برای دیتای جدید عملکرد خوبی نداشته باشد. (مثلا نویز این دیتای خاص را هم یاد گرفته باشد و برای دیتای جدید با نویز متفاوت اشتباه عمل کند.) برای همین قسمتی از دیتا را تحت عنوان دیتای تست از دیتای آموزش جدا می کنند و با آن دیتای متفاوت نیز عملکرد دستگاه را ارزیابی می کنند.

* 1. بخش چهارم

از روش SGDClassifier برای طبقه بندی استفاده شد.

پس از import کردن موارد لازم، جهت رسم تابع هزینه یک کلاس به نام newSGD تعریف می کنیم که از کلاس SGDClassifier ارث بری می کند. دو متد \_\_init\_\_ (مقداردهی اولیه) و partial\_fit (مانند همان partial\_fit کلاس SGDClassifier است با این تفاوت که میانگین مربعات خطا را ذخیره می کند.) را در این کلاس می نویسیم. حال به کمک کلاس تعریف شده و برای 1 تکرار، ماشین را آموزش داده و نمودار تابع اتلاف را رسم می کنیم.



به دلیل سادگی و جدا بودن کلاس ها، این روش و روش قبلی هر دو عملکرد 100 درصدی و مناسبی دارند و نمی توان عملکرد این دو روش را با این دیتاست ساده، چندان مقایسه کرد.

* 1. بخش پنجم

Orange یک نرم‌افزار داده‌کاوی و تحلیل داده متن‌باز و قابل استفاده برای افراد بدون تخصص عمیق در علوم کامپیوتر و داده‌کاوی است. این نرم‌افزار با رابط کاربری گرافیکی (GUI) ساده و کاربرپسند امکانات متنوعی برای بارگیری، تحلیل، و بصری‌سازی داده‌ها ارائه می‌دهد. تعداد زیادی از الگوریتم‌های داده‌کاوی و یادگیری ماشین، ابزارهای پردازش داده، و ابزارهای بصری‌سازی اطلاعاتی در این نرم‌افزار گنجانده شده است.

بعضی از قابلیت‌های مهم Orange عبارتند از:

1. تحلیل داده‌های گوناگون: از جمله تحلیل داده‌های دسته‌ای، رگرسیون، خوشه‌بندی، کاوش جامعه، و پیش‌بینی.

2. تجزیه و تحلیل مرورگری: این ابزار به شما امکان می‌دهد تا با استفاده از محیط گرافیکی، اقدام به تجزیه و تحلیل داده‌ها کنید و نتایج خود را به راحتی بصری‌سازی کنید.

3. پشتیبانی از الگوریتم‌های متنوع: الگوریتم‌های متنوعی از جمله درخت تصمیم، ماشین بردار پشتیبان، شبکه‌های عصبی، و الگوریتم‌های خوشه‌بندی در Orange قابل دسترسی هستند.

4. قابلیت ترکیب ماژول‌ها: کاربران می‌توانند ماژول‌های مختلف را با یکدیگر ترکیب کرده و فرایندهای تحلیلی پیچیده‌تر را ایجاد کنند.

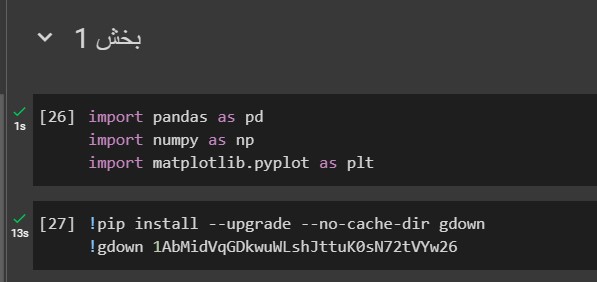
5. پشتیبانی از زبان برنامه‌نویسی Python: کاربران می‌توانند با استفاده از زبان برنامه‌نویسی Python، قابلیت‌های Orange را گسترش داده و سفارشی‌سازی‌های مورد نیاز خود را انجام دهند.

6. پشتیبانی از ویژگی‌های بصری‌سازی: Orange امکانات گسترده‌ای برای بصری‌سازی داده‌ها از جمله نمودارهای مختلف، نمایش‌های چندبعدی، و نمایش‌های شبکه‌ای فراهم می‌کند.

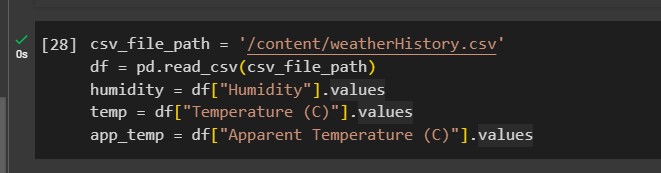
در کل، Orange یک ابزار قدرتمند و کارآمد برای انجام تحلیل داده‌ها به صورت گرافیکی و بدون نیاز به دانش عمیق در زمینه داده‌کاوی است.

1. سوال سوم
   1. بخش اول

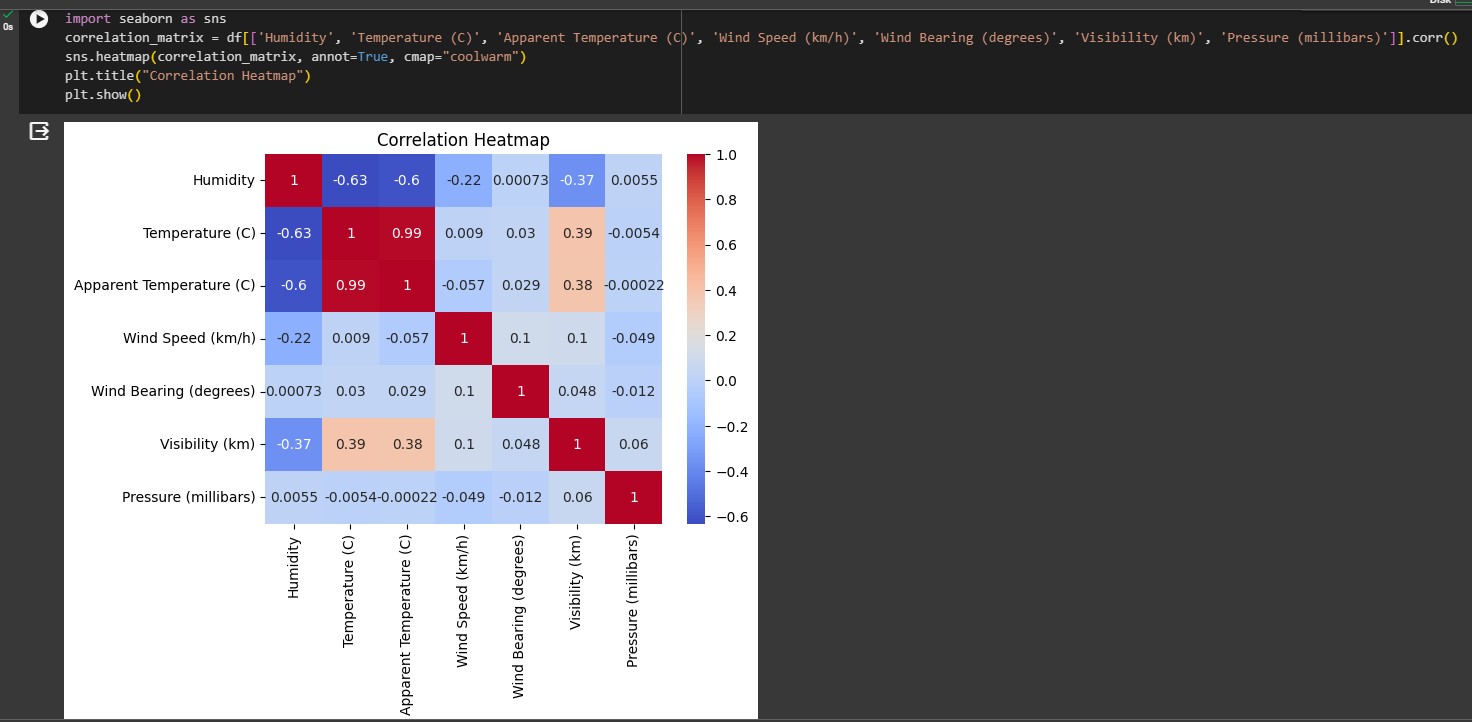
ابتدا موارد مورد نیاز را نصب می کنیم.



سپس پس از نصب gdown، با استفاده از آن دیتاست cvs که دانلود کرده و در drive به صورت public قرار داده ایم را به google colab می آوریم و دیتای مربوط به humidity، Temperature و Apparent Temperature را جدا می کنیم.

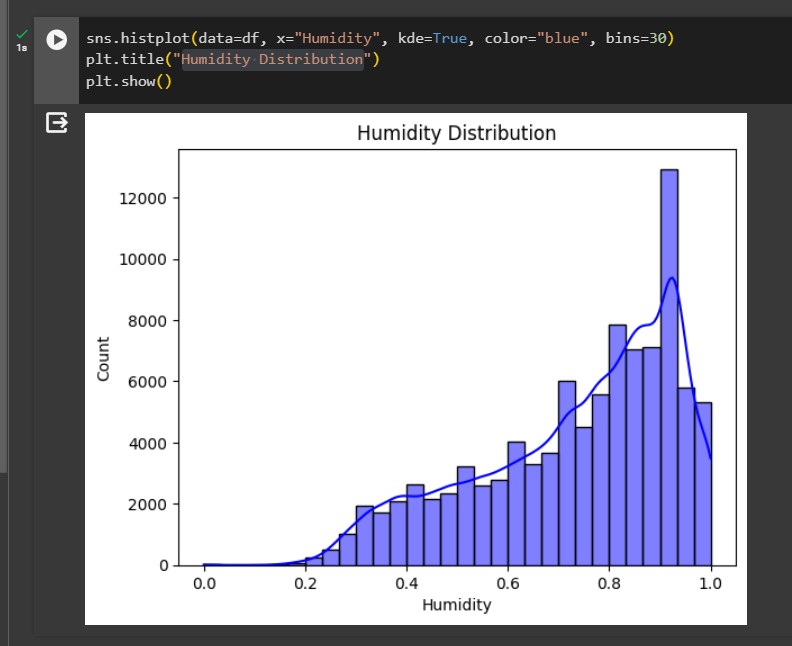


هیت مپ ماتریس همبستگی ویژگی ها

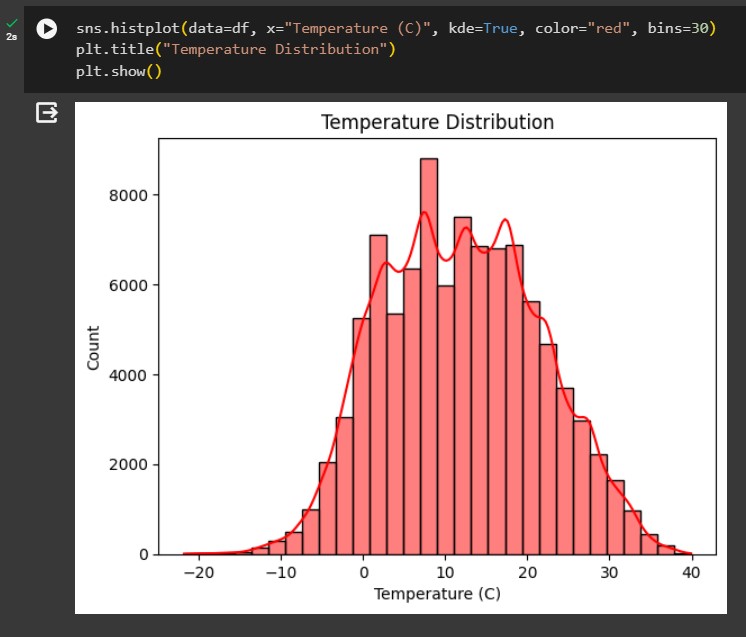


با توجه به این ماتریس مشخص می شود که کدام ویژگی ها بیشترین همبستگی را با یکدیگر دارند و می توانند برای پیشبینی مورد استفاده قرار گیرند. مثلا برای پیشبینی Temperature به ترتیب بهترین ویژگی، Apparent Temperature، Humidity و Visibility است.

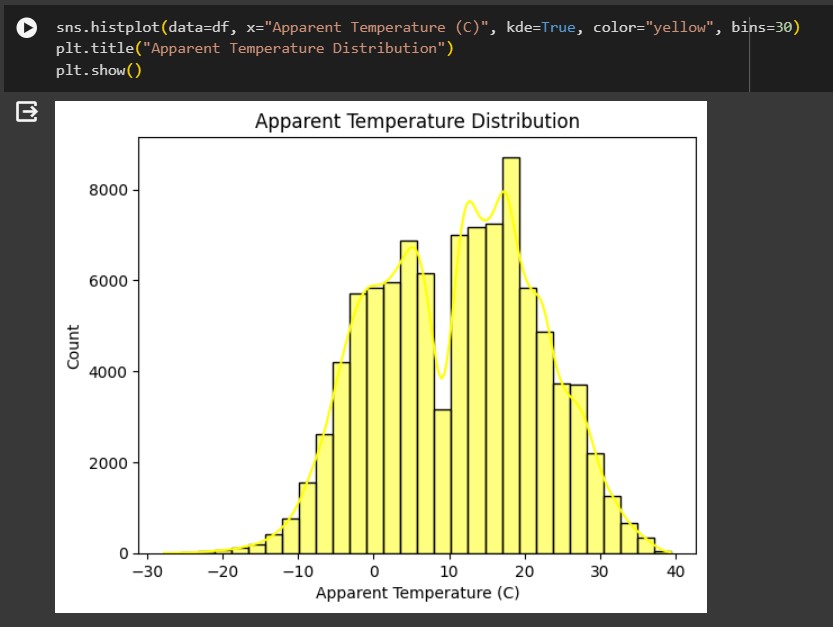
نمودار توزیع رطوبت



نمودار توزیع دما

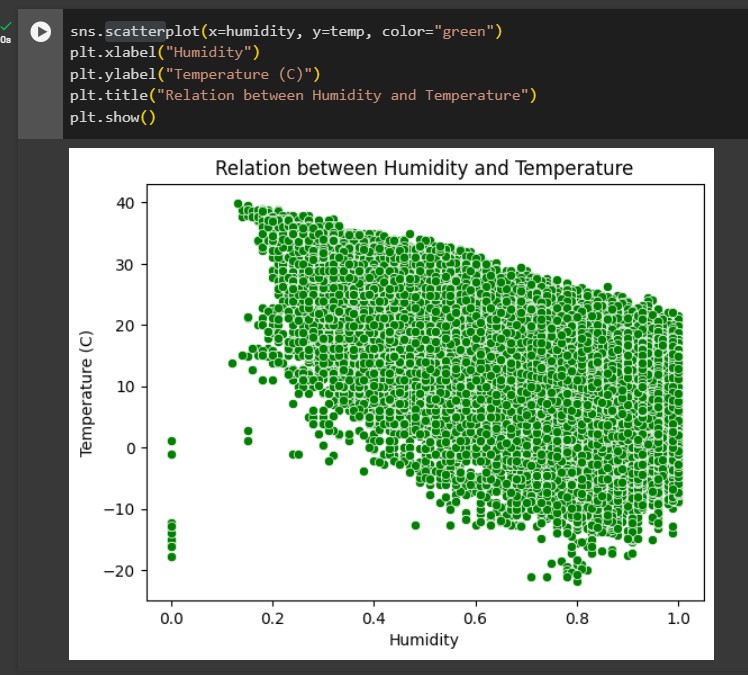


نمودار توزیع دمای ظاهری



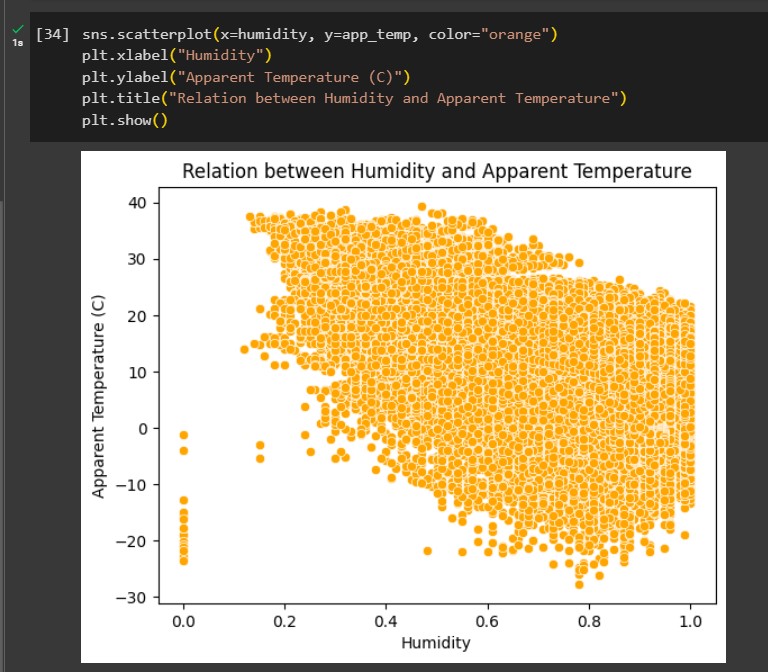
توزیع سه ویژگی در نمودارهای بالا نمایش داده شده است.

نمودار پراکندگی دما برحسب رطوبت



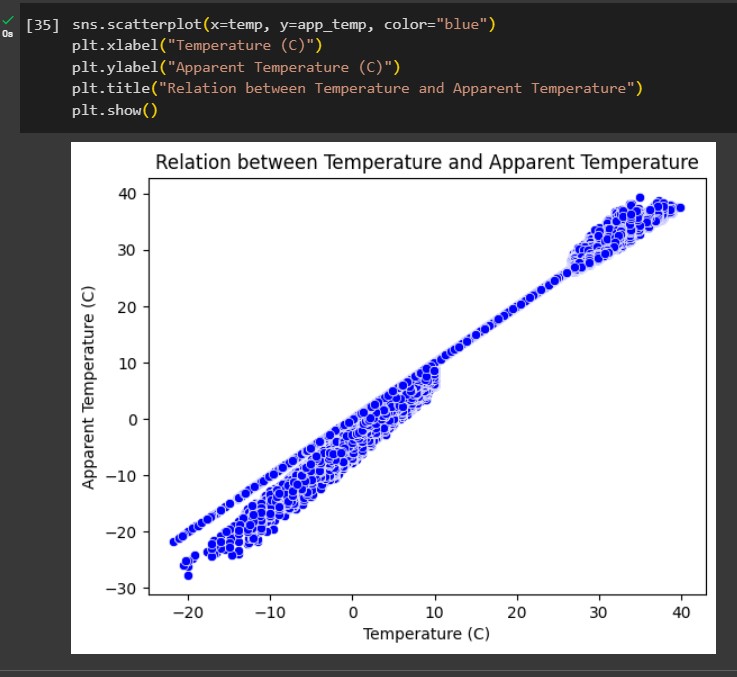
ارتباط خطی تنگاتنگی بین این دو ویژگی مشاهده نمی شود.

نمودار پراکندگی دمای ظاهری برحسب رطوبت



ارتباط خطی تنگاتنگی بین این دو ویژگی مشاهده نمی شود.

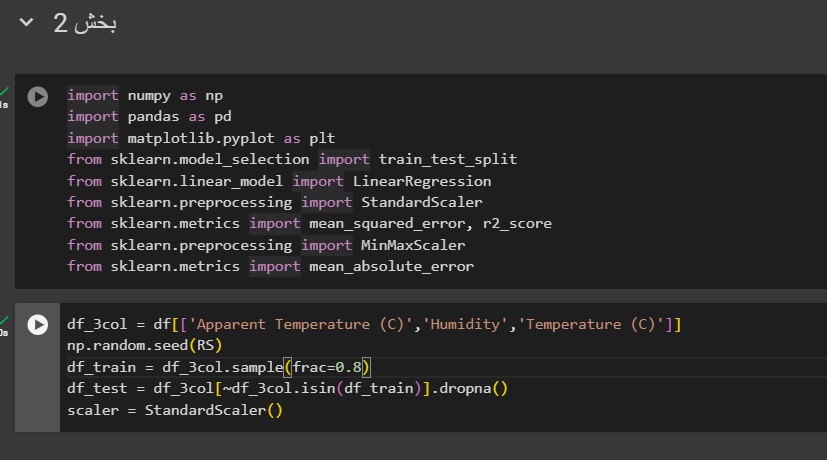
نمودار پراکندگی دمای ظاهری بر حسب دما



ارتباط خطی تنگاتنگی بین این دو ویژگی مشاهده می شود. پس توقع ما این است که با پیشبینی دمای ظاهری بر حسب دما، مقدار خطای کوچکی داشته باشیم.

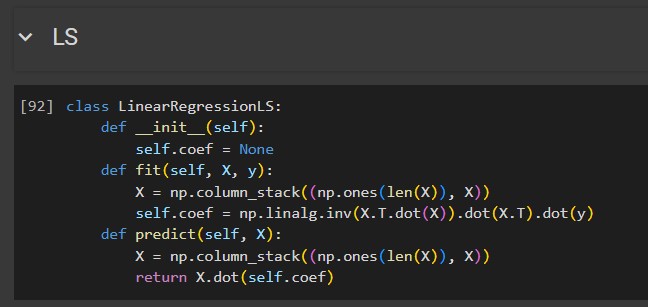
* 1. بخش دوم

ابتدا موارد مورد نیاز را import می کنیم. دیتای دمای ظاهری، دما و رطوبت را از بقیه جدا می کنیم. به مانند سوالات قبل، 20 درصد دیتا را برای تست و 80 درصد را برای آموزش جدا می کنیم.



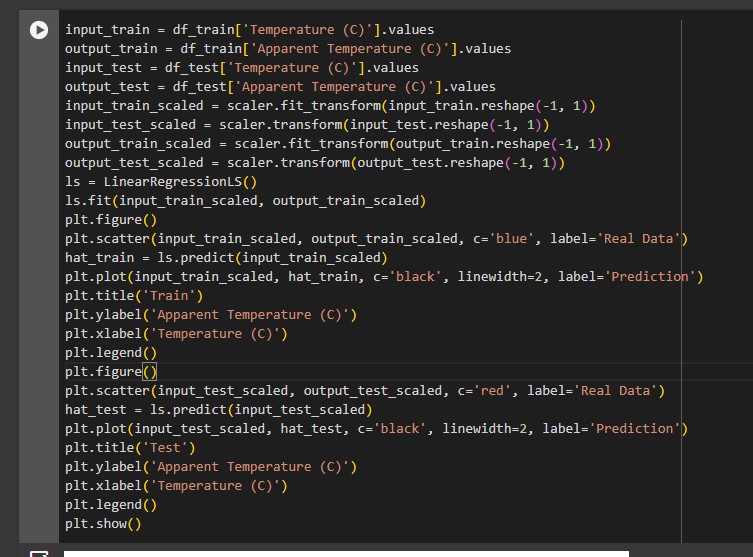
**LS**

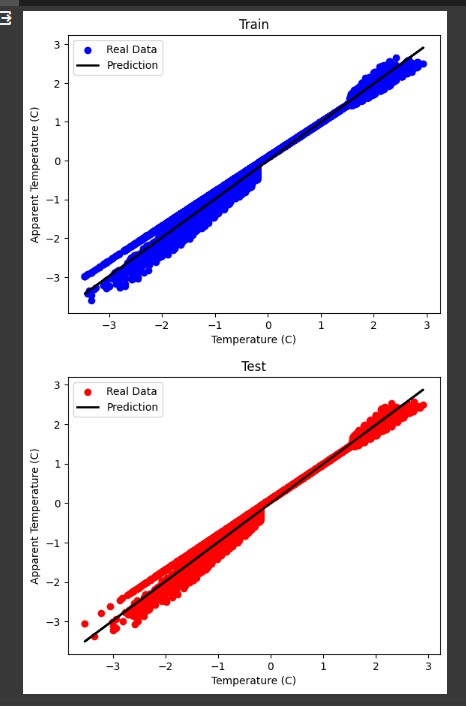
حال کلاس LinearRegressionLS را برای تخمین LS می نویسیم. این کلاس حاوی سه متد \_\_init\_\_ (مقداردهی اولیه)، fit (آموزش ماشین و آپدیت ضرایب) و predict (پیشبینی خروجی بر حسب ورودی وارد شده و ضرایب آموزش یافته) است.



در این قسمت دما را به عنوان ورودی و دمای ظاهری را به عنوان خروجی در نظر می گیریم.

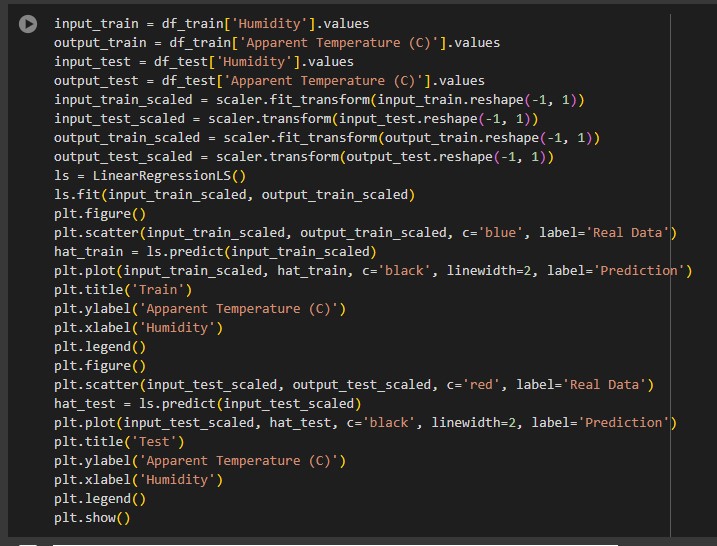
ابتدا دیتا را با StandardScaler، scale می کنیم. سپس به کمک کلاسی که تعریف کردیم، یک مدل یادگیری ماشین آموزش می دهیم. سپس نمودار پراکندگی و خط پیشبینی را رسم می کنیم. این کار را برای دیتای آموزش و تست انجام می دهیم.

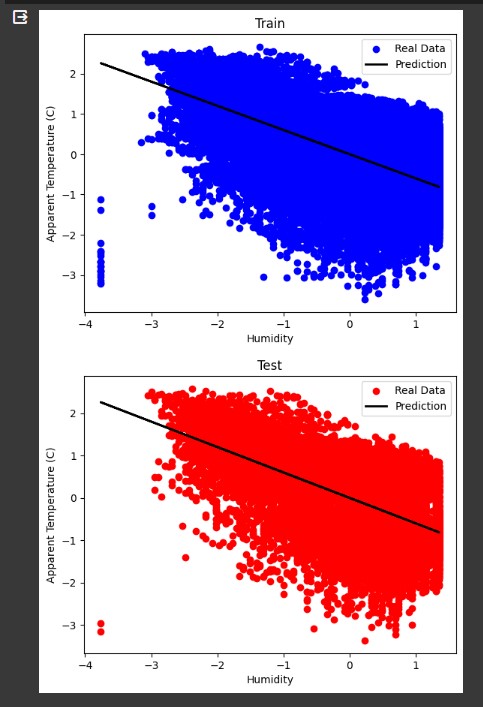




مطابق انتظار به علت وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی دارد.

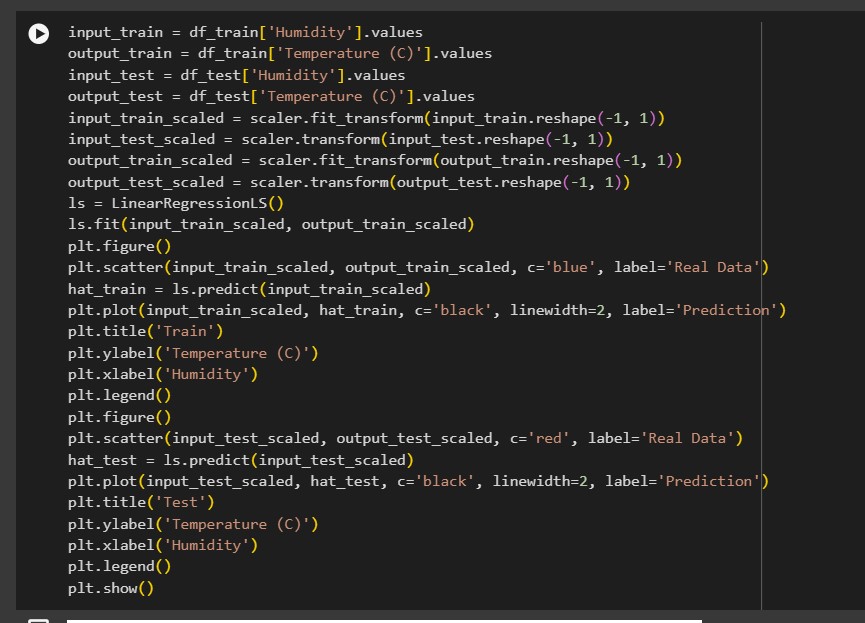
این قسمت شبیه حالت قبل است با این تفاوت که رطوبت را ورودی و دمای ظاهری را به عنوان خروجی درنظر می گیریم.

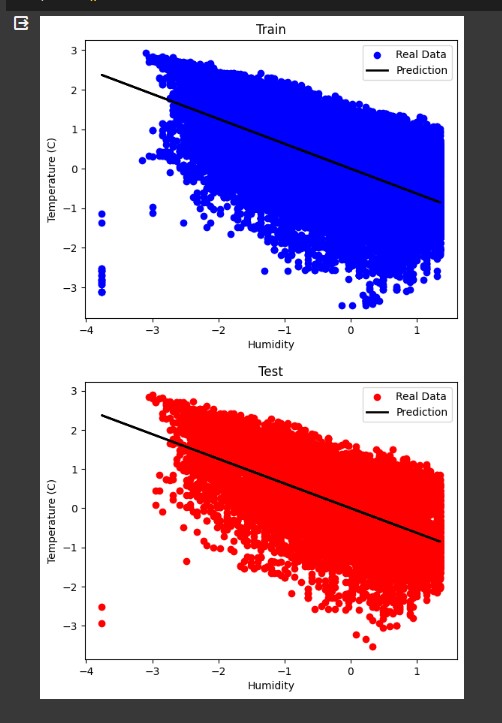




مطابق انتظار به علت عدم وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی ندارد.

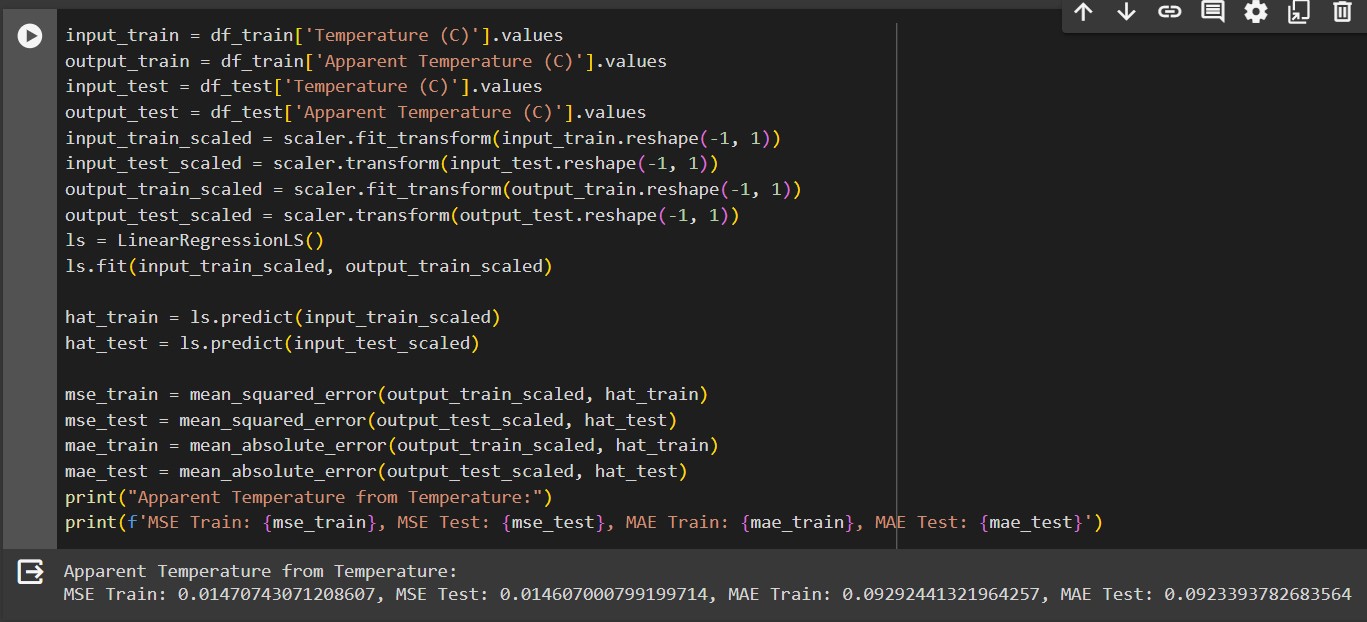
این قسمت شبیه حالت قبل است با این تفاوت که رطوبت را ورودی و دما را به عنوان خروجی درنظر می گیریم.

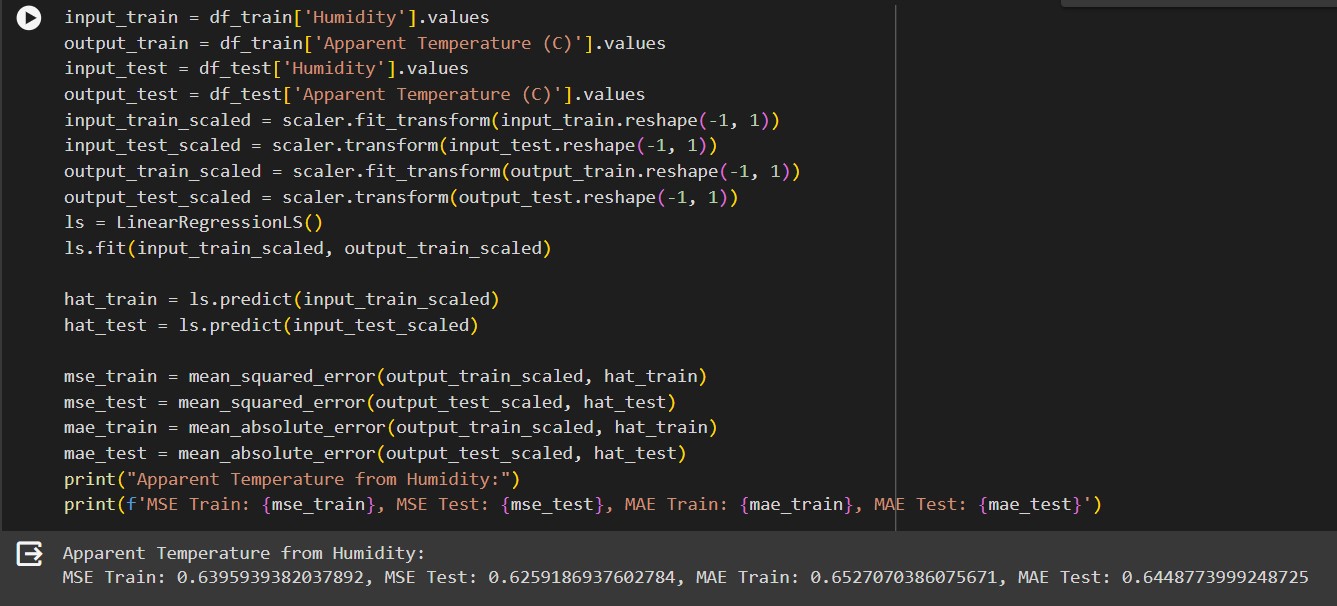




مطابق انتظار به علت عدم وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی ندارد.

برای هر سه حالت بالا، خطای mean squared error و mean absolute error را برای دیتای test و train محاسبه می کنیم. مطابق انتظار کمترین خطا برای حالت اول (پیشبینی دمای ظاهری به کمک دمای واقعی) است.

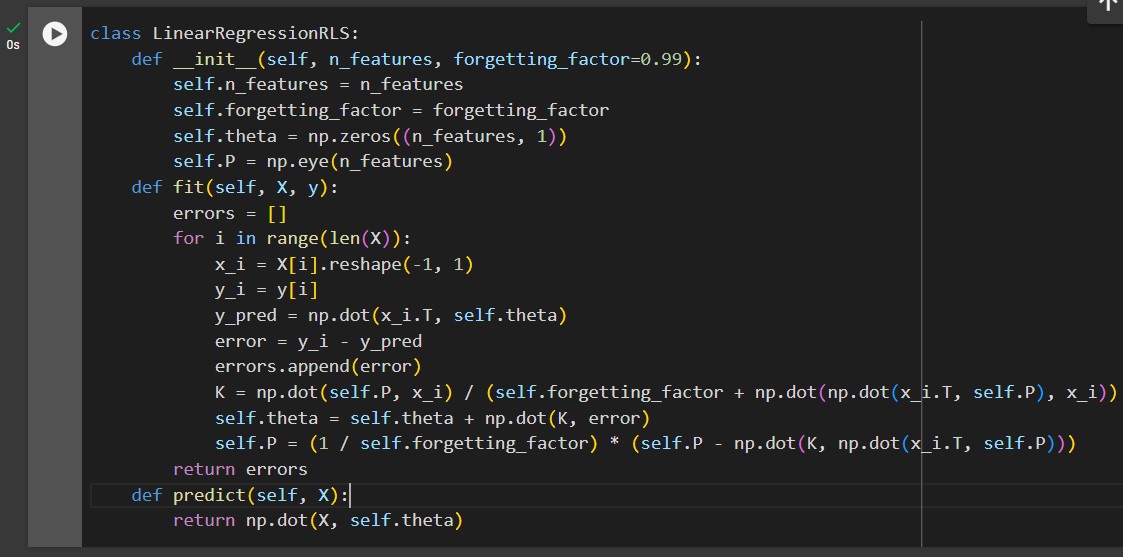






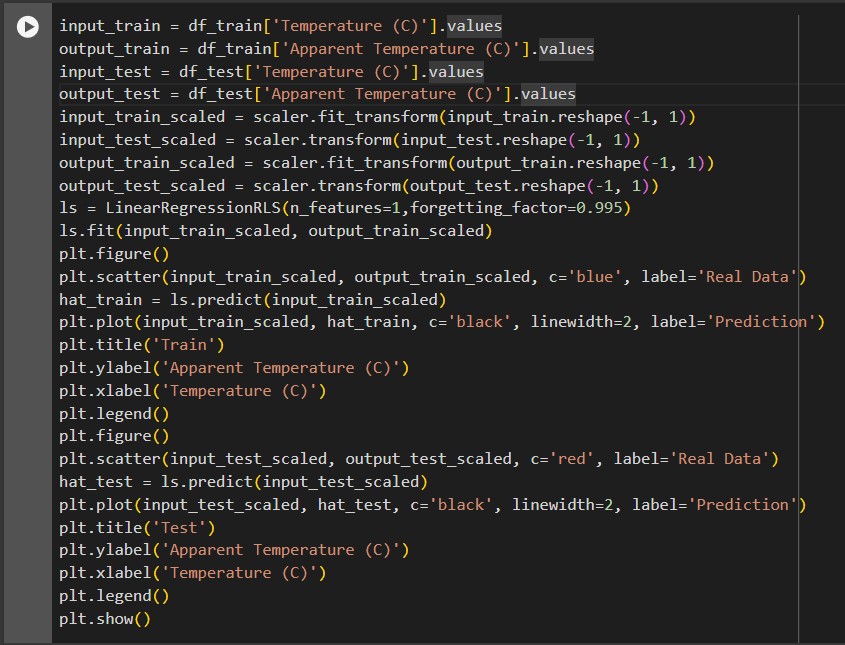
**RLS**

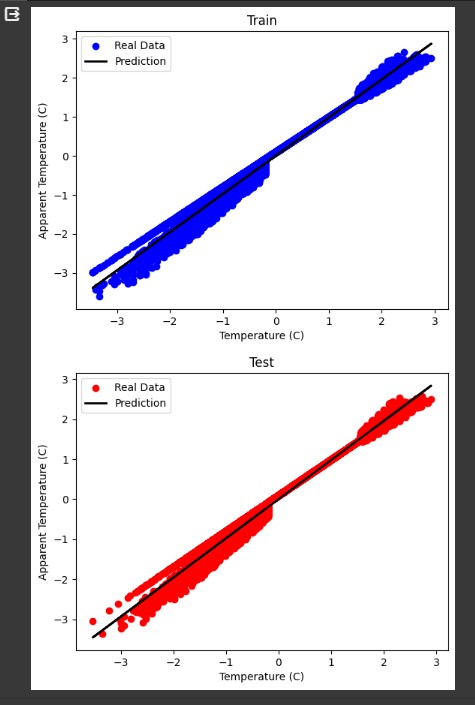
حال کلاس LinearRegressionRLS را برای تخمین RLS می نویسیم. این کلاس حاوی سه متد \_\_init\_\_ (مقداردهی اولیه)، fit (آموزش ماشین و آپدیت ضرایب) و predict (پیشبینی خروجی بر حسب ورودی وارد شده و ضرایب آموزش یافته) است.



در این قسمت دما را به عنوان ورودی و دمای ظاهری را به عنوان خروجی در نظر می گیریم.

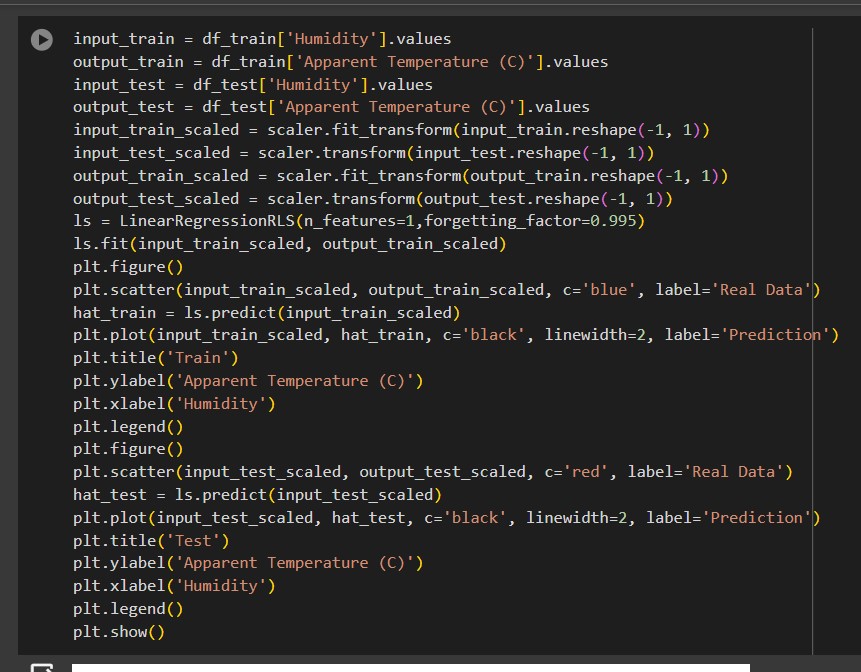
ابتدا دیتا را با StandardScaler، scale می کنیم. سپس به کمک کلاسی که تعریف کردیم، یک مدل یادگیری ماشین آموزش می دهیم. سپس نمودار پراکندگی و خط پیشبینی را رسم می کنیم. این کار را برای دیتای آموزش و تست انجام می دهیم.

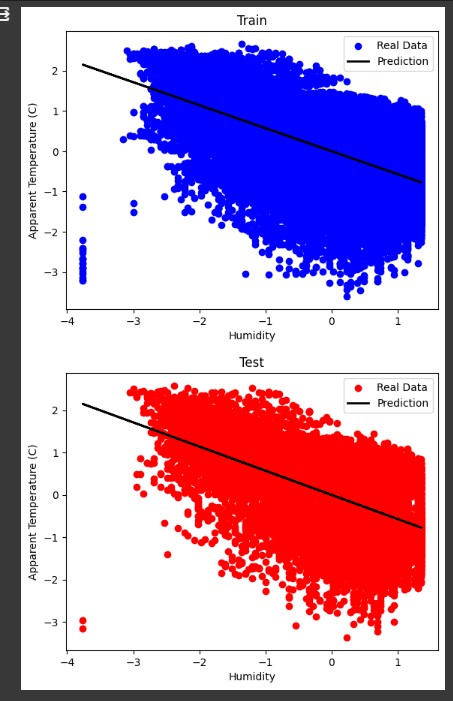




مطابق انتظار به علت وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی دارد.

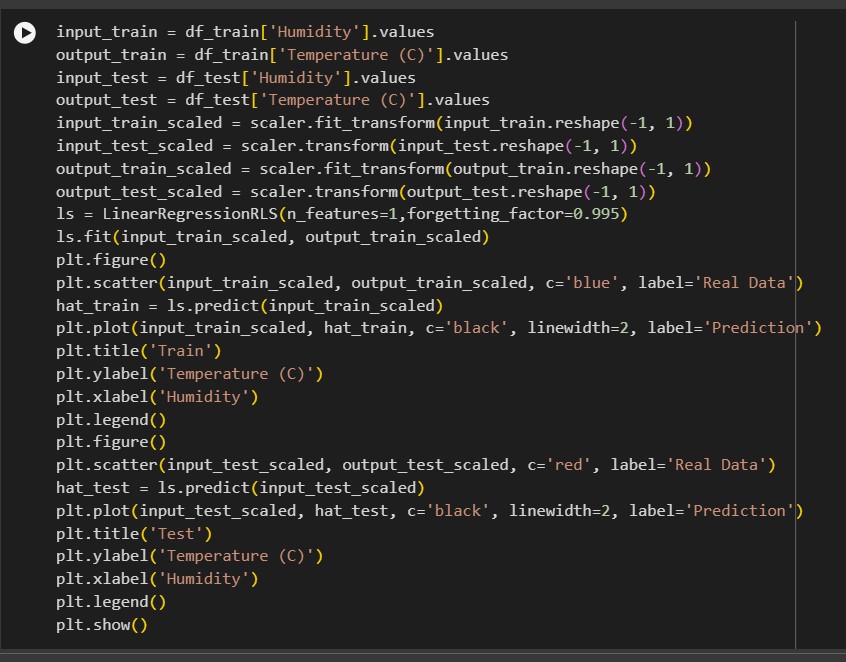
این قسمت شبیه حالت قبل است با این تفاوت که رطوبت را ورودی و دمای ظاهری را به عنوان خروجی درنظر می گیریم.

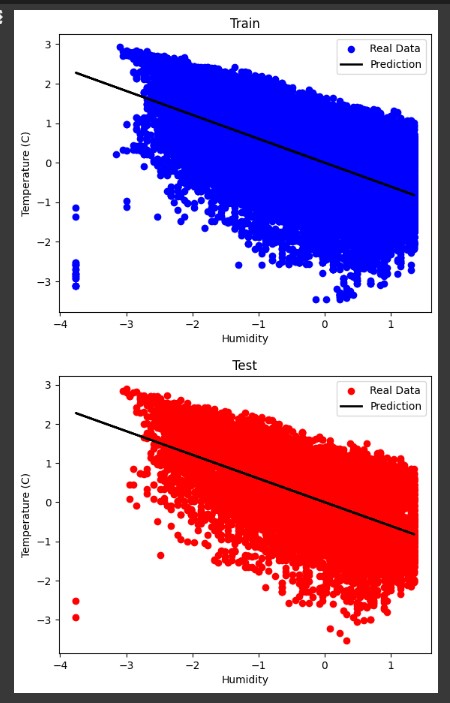




مطابق انتظار به علت عدم وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی ندارد.

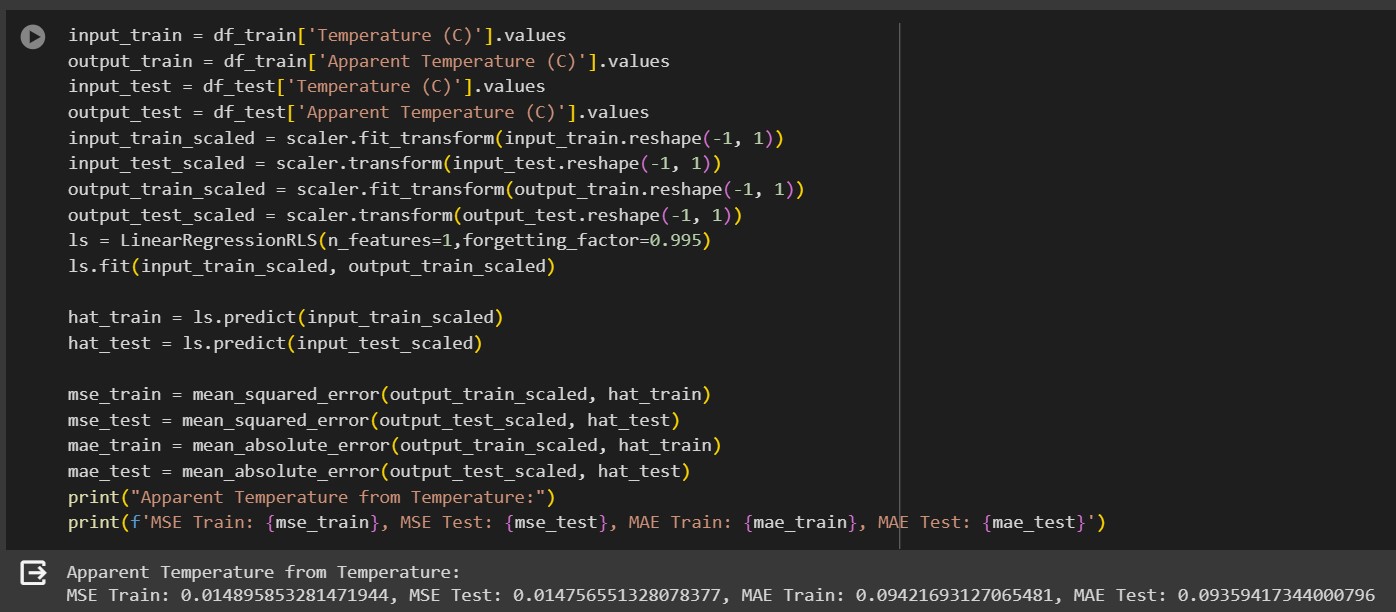
این قسمت شبیه حالت قبل است با این تفاوت که رطوبت را ورودی و دما را به عنوان خروجی درنظر می گیریم.

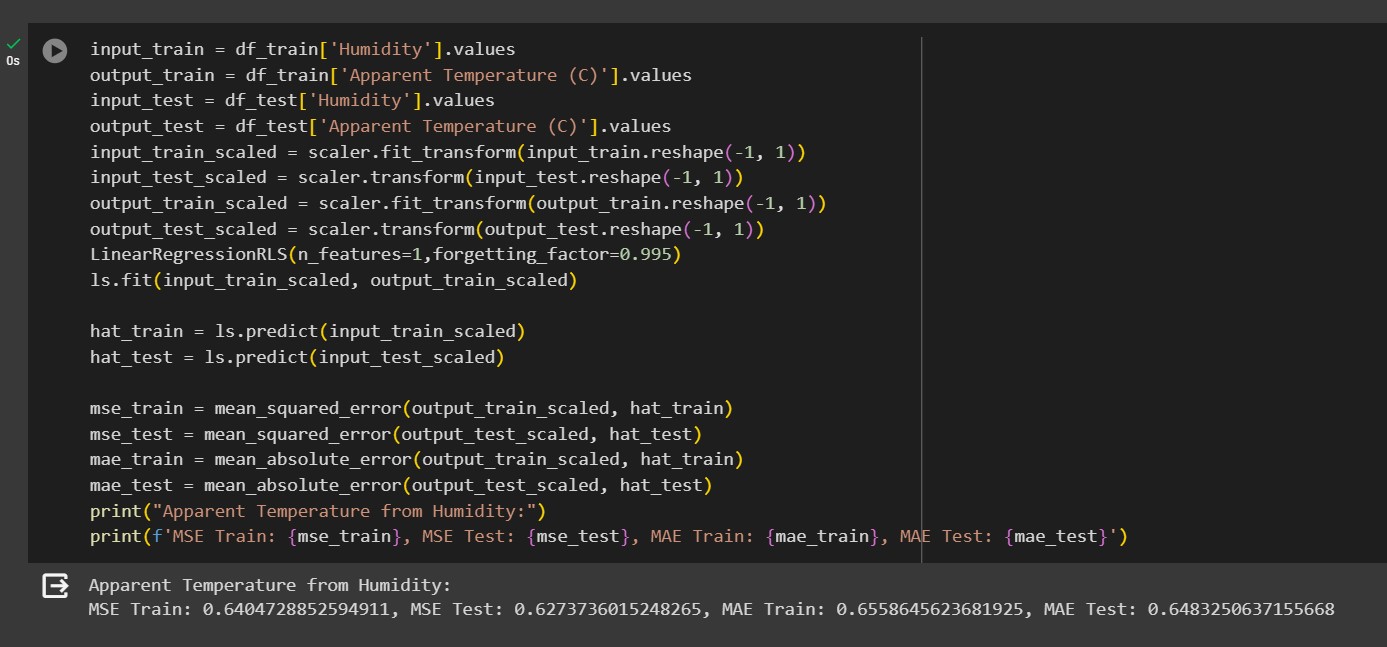


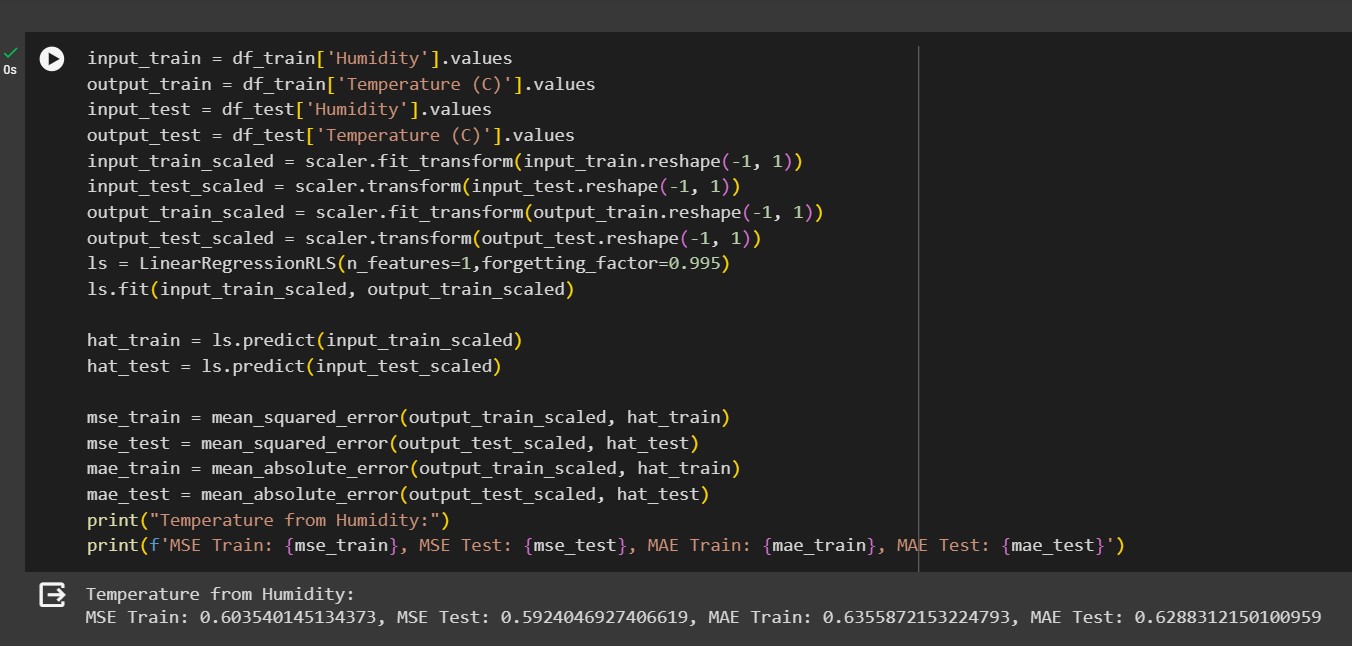


مطابق انتظار به علت عدم وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی ندارد.

برای هر سه حالت بالا، خطای mean squared error و mean absolute error را برای دیتای test و train محاسبه می کنیم. مطابق انتظار کمترین خطا برای حالت اول (پیشبینی دمای ظاهری به کمک دمای واقعی) است.







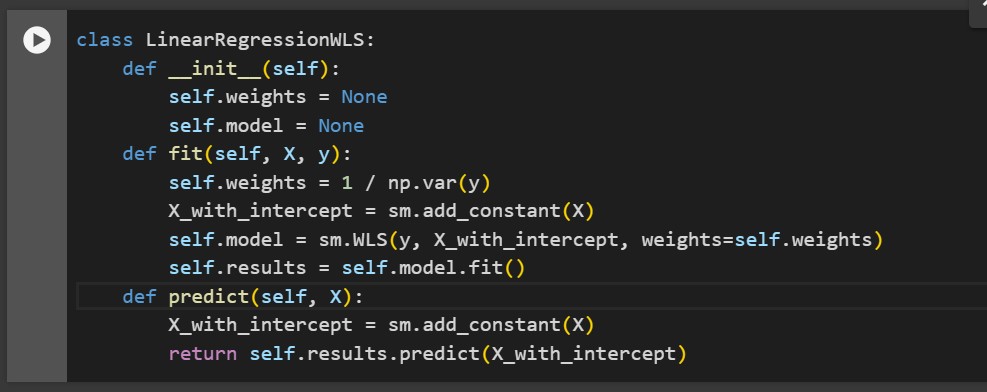
همانطور که مشاهده می شود، برای این دیتاست عملکرد LS و RLS تفاوت چندانی ندارند.

* 1. بخش سوم

روش رگرسیون خطی Weighted Least Squares (WLS) یک روش است که در آن برای تخمین پارامترهای مدل خطی، از یک ماتریس وزنی برای نمونه‌ها استفاده می‌شود. این ماتریس وزنی نشان دهنده اهمیت نمونه‌ها در مدل‌سازی است، به این معنی که نمونه‌های با وزن بیشتر در محاسبه تخمین‌ها واردتر هستند و وزن کمتری به نمونه‌های کم‌اهمیت تخصیص داده می‌شود.

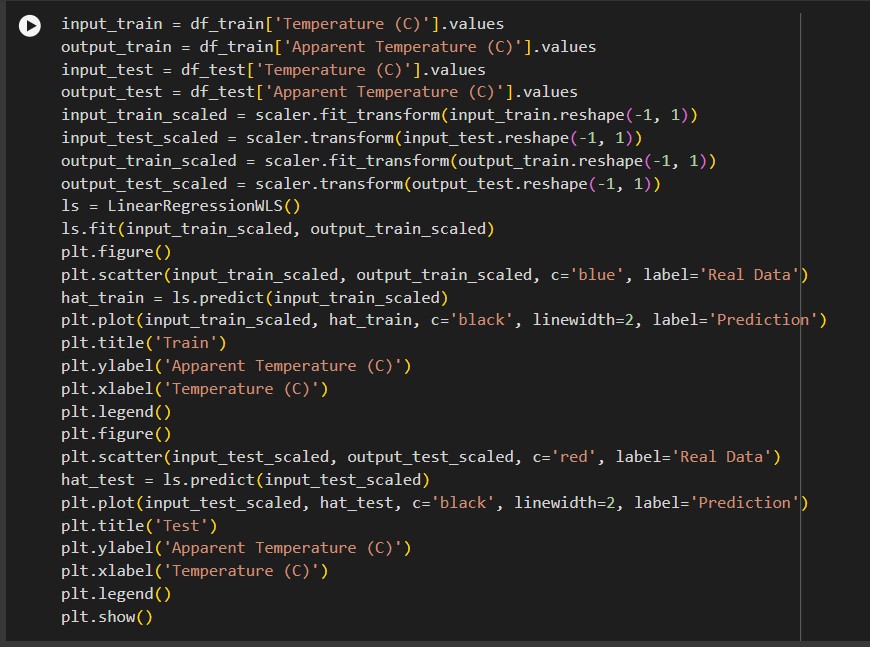
بر خلاف روش عادی رگرسیون خطی Least Squares (LS)، در WLS وزن مخصوص به هر نقطه در مدل‌سازی استفاده می‌شود. این وزن‌ها معمولاً بر اساس نوع مشکل و داده‌ها تعیین می‌شوند. به عنوان مثال، اگر نمونه‌ها با ویژگی‌های کیفیت بالاتر دارای وزن بیشتر باشند (به عنوان مثال، نمونه‌هایی که دقیق‌تر اندازه گرفته شده‌اند)، آن‌ها به طور معمول وزن بیشتری در مدل‌سازی خواهند داشت. فرق اصلی بین روش WLS و LS در استفاده از ماتریس وزنی است. در روش LS، همه نمونه‌ها به یکسان در مدل‌سازی استفاده می‌شوند، در حالی که در WLS، وزن‌ها برای هر نمونه ممکن است متفاوت باشند و به نمونه‌های با اهمیت بیشتر وزن بیشتری تخصیص داده شود. این امر به مدل کمک می‌کند تا بهتر به داده‌های با کیفیت بالا و حساس به نویز پاسخ دهد و از اغتشاشات ناشی از داده‌های با کیفیت پایین کاسته شود.

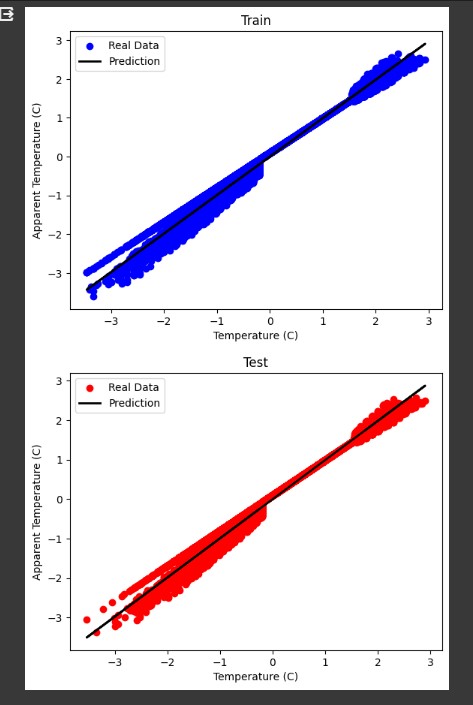
حال کلاس LinearRegressionWLS را برای تخمین WLS می نویسیم. این کلاس حاوی سه متد \_\_init\_\_ (مقداردهی اولیه)، fit (آموزش ماشین و آپدیت ضرایب) و predict (پیشبینی خروجی بر حسب ورودی وارد شده و ضرایب آموزش یافته) است.



در این قسمت دما را به عنوان ورودی و دمای ظاهری را به عنوان خروجی در نظر می گیریم.

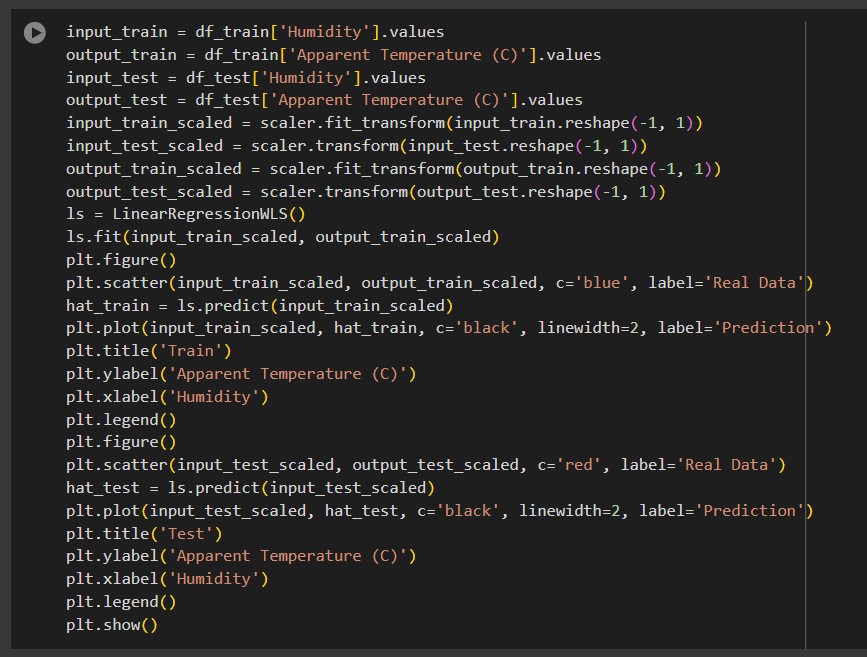
ابتدا دیتا را با StandardScaler، scale می کنیم. سپس به کمک کلاسی که تعریف کردیم، یک مدل یادگیری ماشین آموزش می دهیم. سپس نمودار پراکندگی و خط پیشبینی را رسم می کنیم. این کار را برای دیتای آموزش و تست انجام می دهیم.

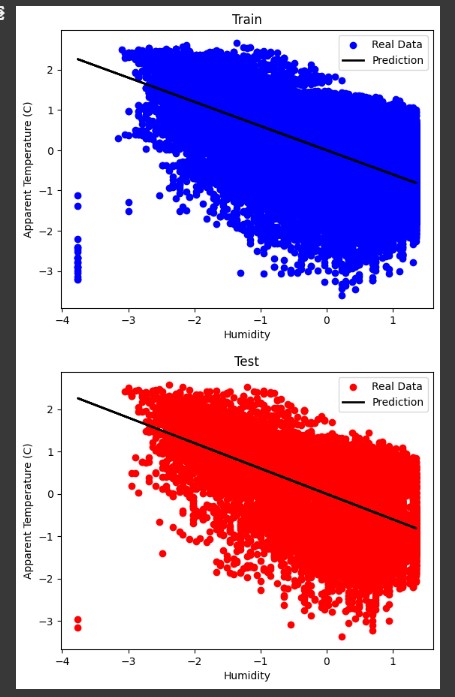




مطابق انتظار به علت وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی دارد.

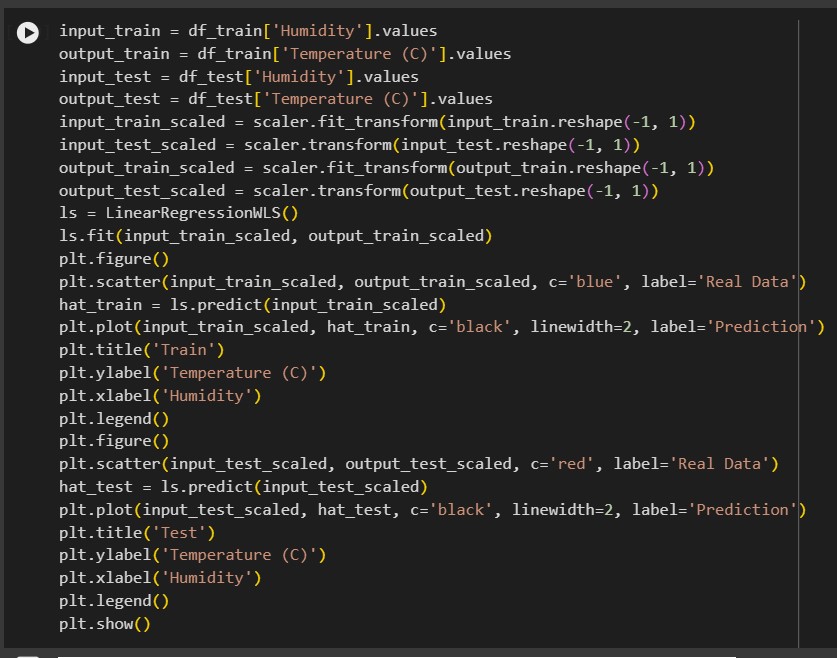
این قسمت شبیه حالت قبل است با این تفاوت که رطوبت را ورودی و دمای ظاهری را به عنوان خروجی درنظر می گیریم.

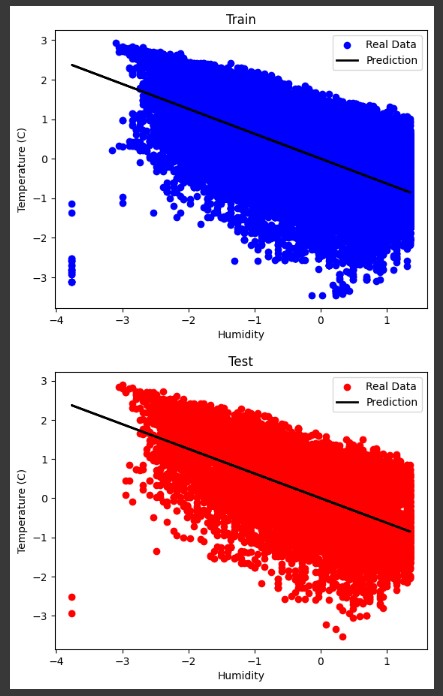




مطابق انتظار به علت عدم وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی ندارد.

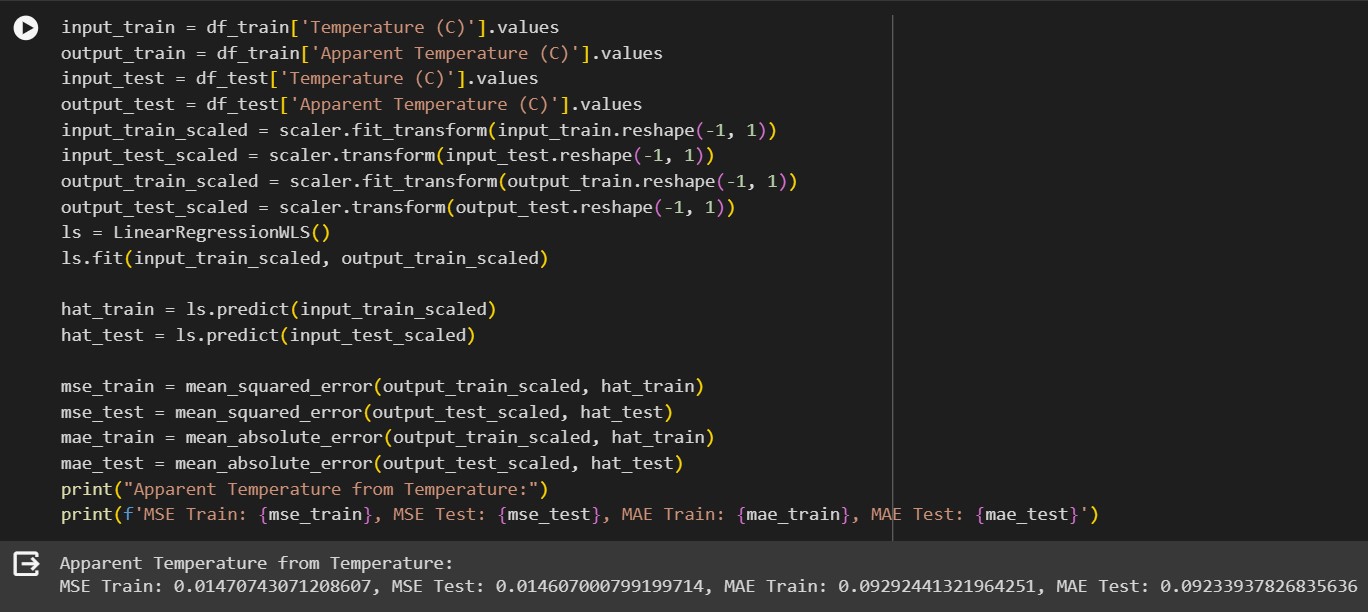
این قسمت شبیه حالت قبل است با این تفاوت که رطوبت را ورودی و دما را به عنوان خروجی درنظر می گیریم.

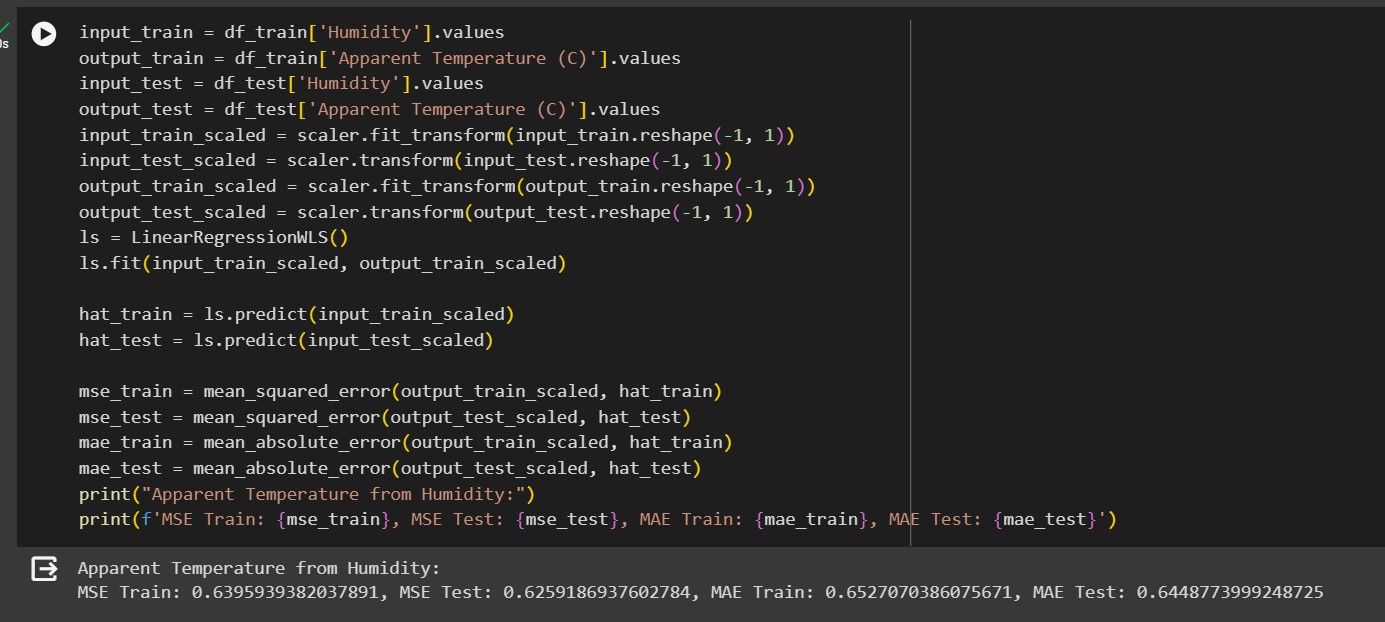


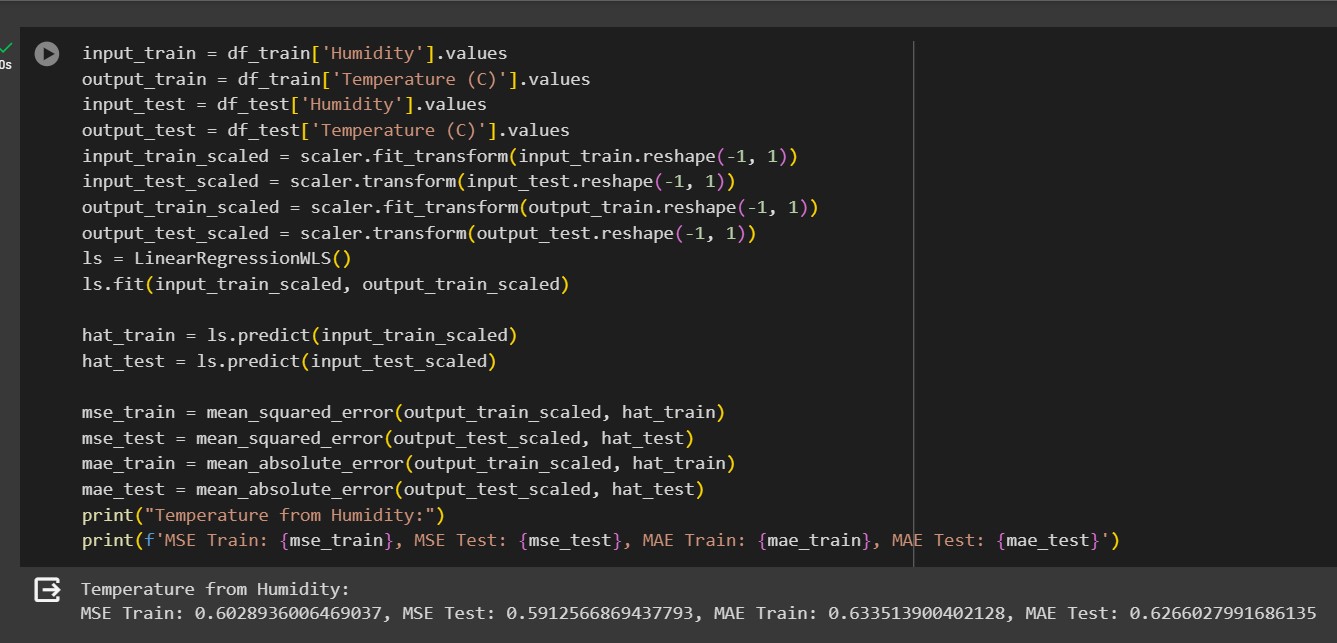


مطابق انتظار به علت عدم وجود همبستگی بالا بین دو ویژگی، خط پیشبینی تطابق خوبی روی دیتای واقعی ندارد.

برای هر سه حالت بالا، خطای mean squared error و mean absolute error را برای دیتای test و train محاسبه می کنیم. مطابق انتظار کمترین خطا برای حالت اول (پیشبینی دمای ظاهری به کمک دمای واقعی) است.







* 1. بخش چهارم

الگوریتم QR-Decomposition-Based Recursive Least Squares (RLS) یک روش برای حل مسائل رگرسیون خطی است که بر پایه تجزیه QR ماتریس طراحی شده است. این الگوریتم مختص به مسائلی است که تعداد متغیرهای مستقل بسیار زیاد است و انتظار می‌رود که مدل به طور پویا با تغییرات در داده‌ها بروزرسانی شود.

در الگوریتم QR-Decomposition-Based RLS، ماتریس مربعی X از ویژگی‌ها را به دو ماتریس Q و R تجزیه می‌کنیم، به طوری که X = QR باشد. سپس از این تجزیه برای به‌روزرسانی پارامترهای مدل استفاده می‌شود.

با استفاده از این تجزیه، مسئله رگرسیون خطی به دو مرحله تقسیم می‌شود:

۱. مرحله قدرت: ماتریس R به‌دست آمده از تجزیه QR می‌تواند به طور مستقیم با استفاده از روش‌های متعارف مسائل رگرسیون خطی حل شود، بدون نیاز به بازگشت به عقب.

۲. مرحله بازگشت به عقب: پس از به‌روزرسانی پارامترهای مدل با استفاده از ماتریس R، ماتریس Q به‌روزرسانی می‌شود تا اطلاعات جدید وارد به مدل شود.

تفاوت اصلی بین الگوریتم QR-Decomposition-Based RLS و RLS ساده در استفاده از تجزیه QR ماتریس است. در الگوریتم ساده RLS، محاسبات به‌روزرسانی پارامترهای مدل براساس ماتریس غیرقابل تجزیه XTX انجام می‌شود. این ماتریس می‌تواند به دلیل بزرگ بودن تعداد ویژگی‌ها و نمونه‌ها باعث محاسبات سنگین شود. اما با استفاده از تجزیه QR، این مشکل کاهش می‌یابد زیرا ماتریس R دارای ابعاد کوچکتری است که محاسبات را سریع‌تر واقعی می‌کند و همچنین باعث از بین بردن محاسبات غیرضروری می‌شود.