به نام خدا

1928
K. N. Toosi University of Technology

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش پروژه

شهاب مقدادی نیشابوری

۴۰۰۰۹۴۴۳

استاد : آقای دکتر مهدی علیاری

بهمن ۱۴۰۳

فهرست مطالب

عنوان شماره صفحه

بخش۱: تخمین قیمت با استفاده از MLP و RBF ۴

توضیح دیتاست ۴

تمیز کردن دیتاست ۵

به دست آوردن بهترین بهترین شبکه MLP ساده برای تخمین قیمت ۷

اضافه کردن لایه RBF به شبکه به دست آمده ۱۲

استفاده از مدل فازی ۲۲

بخش۲: تخمین نوع با استفاده از SVM ۲۷

چکیده

در این پروژه، هدف اصلی تحلیل و پیش‌بینی قیمت الماس‌ها بر اساس ویژگی‌های مختلف آن‌ها مانند وزن، ابعاد، رنگ، شفافیت و برش است. با استفاده از روش‌های یادگیری ماشین، مدل‌هایی توسعه داده می‌شوند تا رابطه بین این ویژگی‌ها و قیمت الماس را شناسایی کرده و قیمت را به‌دقت پیش‌بینی کنند. در مرحله بعد، تمرکز بر روی تشخیص نوع الماس (طبیعی یا مصنوعی) قرار می‌گیرد. این بخش از پروژه به شناسایی الگوهایی می‌پردازد که بتوانند با دقت بالا، الماس‌های طبیعی را از مصنوعی تشخیص دهند. در نهایت، این پروژه به بررسی کارایی مدل‌های مختلف یادگیری ماشین در این دو وظیفه و مقایسه نتایج آن‌ها می‌پردازد. این تحلیل‌ها می‌توانند به درک بهتر عوامل موثر بر قیمت و تشخیص نوع الماس کمک کنند و به عنوان ابزاری مفید در صنعت جواهرات مورد استفاده قرار گیرند.

بخش۱: تخمین قیمت با استفاده از MLP و RBF

توضیح دیتاست

در بخش اول دیتاست را به صورت دقیق توضیح میدهیم. این دیتاست حاوی اطلاعات گوناگون از ۶۴۸۵ عدد الماس گوناگون است که بعضی از سطر های آن حاوی اطلاعات NaN میباشد. این اطلاعات شامل:

۱- شکل الماس

۲- کیفیت برش الماس

۳- رنگ الماس

۴- سطح شفافیت الماس

۵- قیراط الماس

۶- نسبت طول به عرض الماس

۷- ضخامت الماس نسبت به عرض آن

۸- عرض سطح بالایی الماس به صورت درصدی نسبت به پهن‌ترین قسمت آن

۹- میزان کیفیت پولیش سطح آن

۱۰- دقت شکل الماس

۱۱- قطر لبه ی الماس

۱۲- اندازه پایین‌ترین سطح الماس

۱۳- طول الماس به میلیمتر

۱۴- عرض الماس به میلیمتر

۱۵-ارتفاع الماس به میلیمتر

۱۶- قیمت الماس(ستون هدف)

۱۷- نوع الماس(ستون هدف)

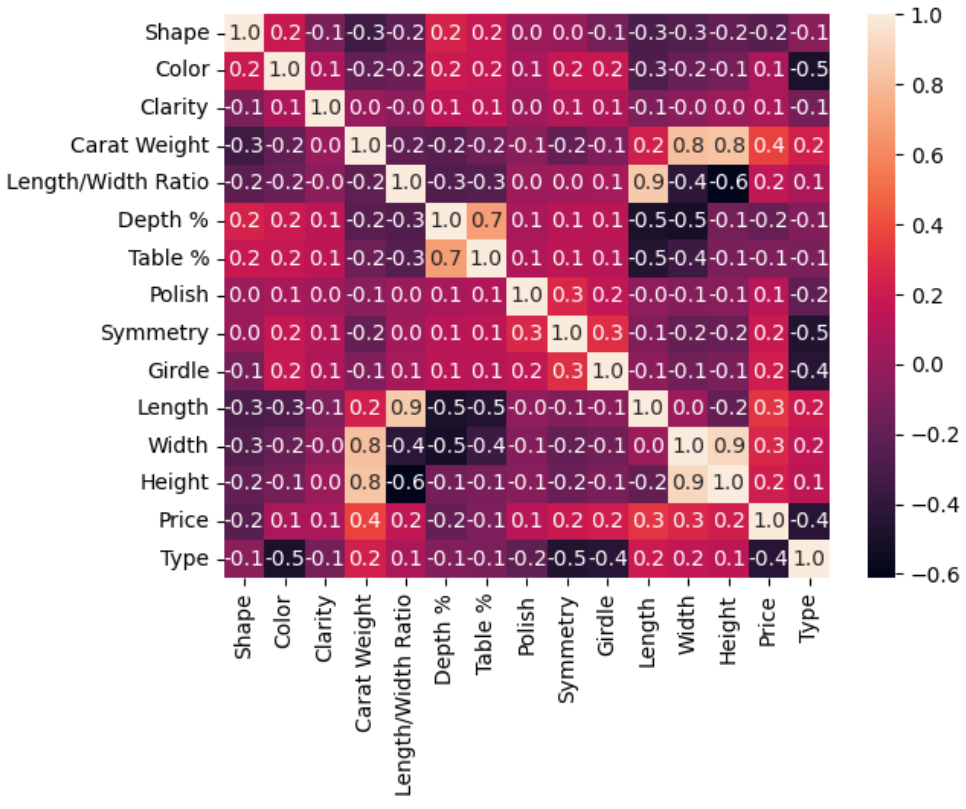
۱۸- میزان نوری که در برابر ماورای بنفش از خود ساطع میکند

میباشد.

تمیز کردن دیتاست

در ابتدا برای تمیز کردن دیتا، سطر های مشابه را حذف میکنیم که با این کار ۳ سطر حذف میشوند. سپس سه سطر cut، culet و Fluorescence را به مرتبط نبودن آنها به فیلد های هدف حذف میکنیم و سپس داده های عددی NaN را با استفاده از ابزار interpolate بازسازی میکنیم. تابع df.interpolate در پانداس برای پر کردن مقادیر گم شده در یک دیتافریم به کار می رود. این تابع با روش های مختلف مانند خطی، چند جمله ای و اسپلاین مقدارهای گمشده را تخمین می زند. در حالت خطی مقدار هر مقدار گم شده بر اساس مقادیر قبلی و بعدی در همان ستون تخمین زده می شود. روش های پیچیده تر مانند اسپلاین از مدل های ریاضی برای پیش بینی مقادیر استفاده می کنند. این تابع به کاربران اجازه می دهد روش مورد نظر خود را انتخاب کنند و میزان دقت پیش بینی را تنظیم کنند. و برای داده های غیر عددی، مقادیر NaN را با تکرارشونده ترین مقدار هر ستون جایگزین میشوند.

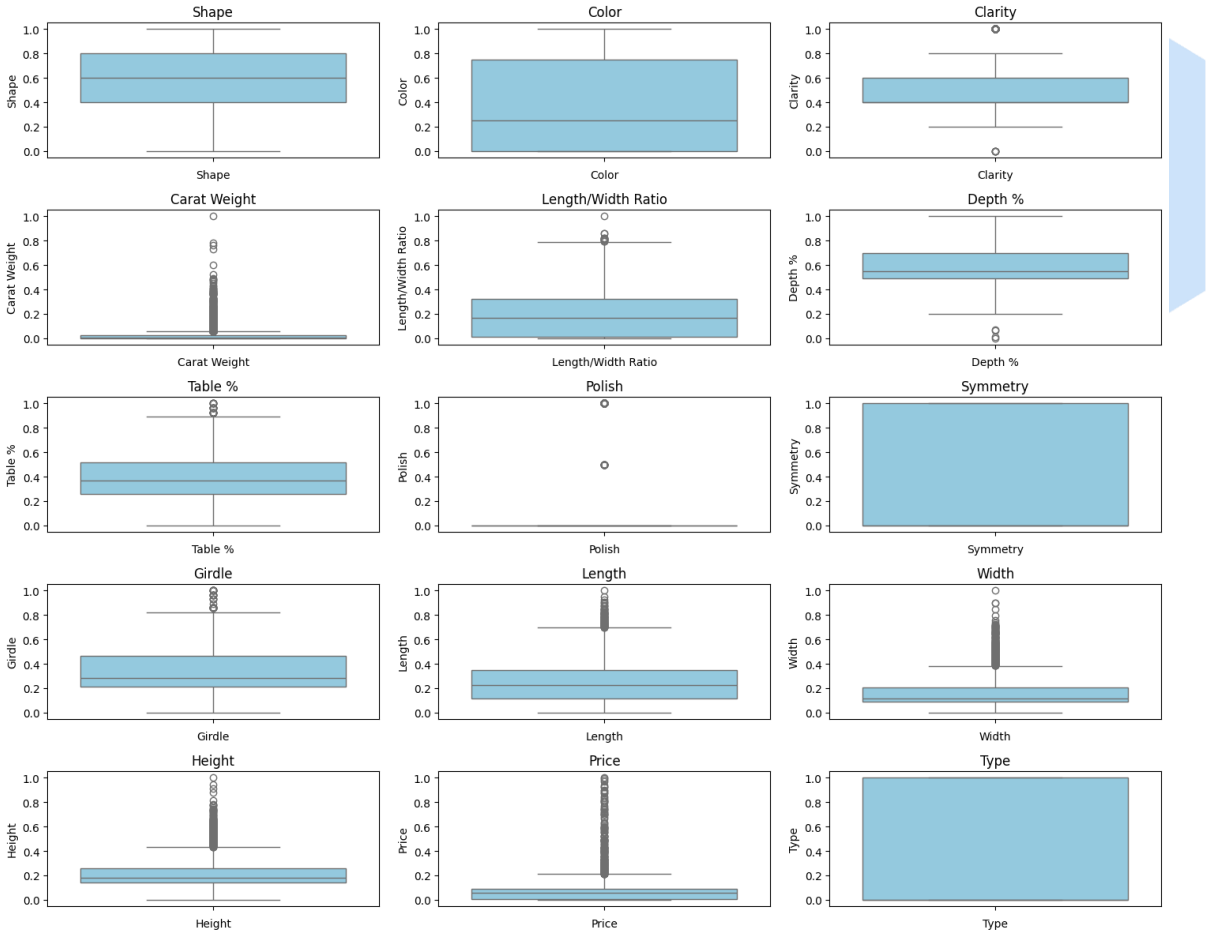
پس از رسیدگی به داده های NaN،به داده های غیر عددی را با کمک LabelEncode مقدار عددی اختصاص میدهیم و به این صورت، بخش اول data cleaning به پایان میرسد. در انتهای این بخش، مقادیر همبستگی پارامتر ها نسبت به هم را مشاهده میکنیم:



همانطور که مشاهده میشود، فیلد قیمت با نوع و قیراط الماس بیشترین مقدار همبستگی را دارد و فیلد نوع الماس با Price، Gridle، Symmetry و Color بیشترین ارتباط را دارد.

MinMaxScalerزمانی بهتر از StandardScaler عمل می کند که داده ها دارای توزیع غیر نرمال باشند یا محدوده مشخصی داشته باشند. این روش مقادیر را در یک بازه ثابت مانند صفر تا یک مقیاس بندی می کند و برای الگوریتم هایی که به محدوده ورودی حساس هستند مانند شبکه های عصبی مناسب تر است. همچنین اگر داده ها دارای مقدارهای پرت زیادی باشند MinMaxScaler کمتر تحت تأثیر آن ها قرار می گیرد زیرا مقیاس بندی را بر اساس حداقل و حداکثر مقادیر انجام می دهد نه میانگین و انحراف معیار. در مقابل اگر داده ها توزیع نرمال داشته باشند یا الگوریتم مورد استفاده به تغییرات واریانس حساس باشد StandardScaler گزینه بهتری خواهد بود. بنابراین برای این دیتاست از متد MinMaxScaler برای نرمال سازی استفاده شده و ستون ها را با این متد نرمالایز میکنیم.

در مرحله بعد توضیع داده ها را مشاهده میکنیم(برای این کار از boxplot استفاده میکنیم):



همانطور که مشاهده میشود، دیتاست به علت نداشتن تمام فیچر هایی که به قیمت نهایی الماس مربوط میشوند، از توضیح اماری خوبی برخوردار نیست و داده های پرت زیادی دارد. برای حل این مشکل، داده های پرت را حذف میکنیم. داده های پرت را از مجموعه داده حذف می کنیم تا کیفیت و دقت مدل بهبود یابد. این کار با استفاده از روشی به نام محدوده بین چارکی انجام می شود که مقادیر خیلی دور از حد معمول را شناسایی کرده و حذف می کنیم. در نتیجه، داده های باقی مانده توزیع متعادل تری دارند و از تأثیر منفی داده های غیرعادی بر روی مدل جلوگیری می شود. برای هر ویژگی مشخص شده، مقادیر چارک اول و سوم را محاسبه میکنیم و از آنها برای تعیین محدوده قابل قبول داده ها استفاده می میکنیم. در نهایت، داده هایی که خارج از این محدوده هستند حذف شده و یک مجموعه داده پاکسازی شده به دست می آید. اما برای نشان دادن تفاوت، هر دو مدل داده را نگه میداریم تا در آینده روش برتر انتخاب شود. سپس از این دیتافریم، داده های ورودی و خروجی های مد نظر استخراج شده و داده های تست با اندازه ۰.۲ داده های ترین جدامیشوند.

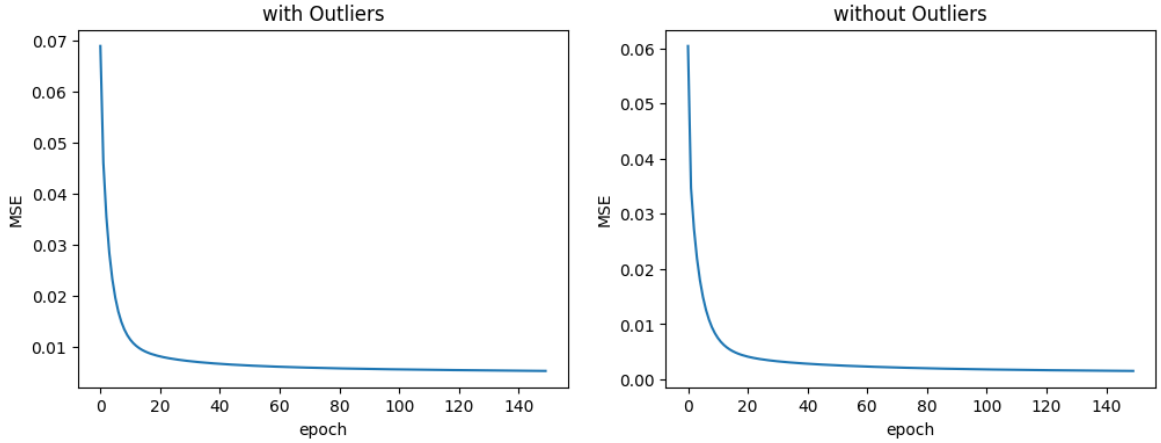
به دست آوردن بهترین بهترین شبکه MLP ساده برای تخمین قیمت

در مرحله اول، برای هر کدام ازدیتاها(دیتای اصلی و پس از حذف داده های پرت) یک مدل ترین میشود که تنها شامل یک عدد لایه پنهان میباشد، معماری این شبکه ها به صورت زیر میباشند:

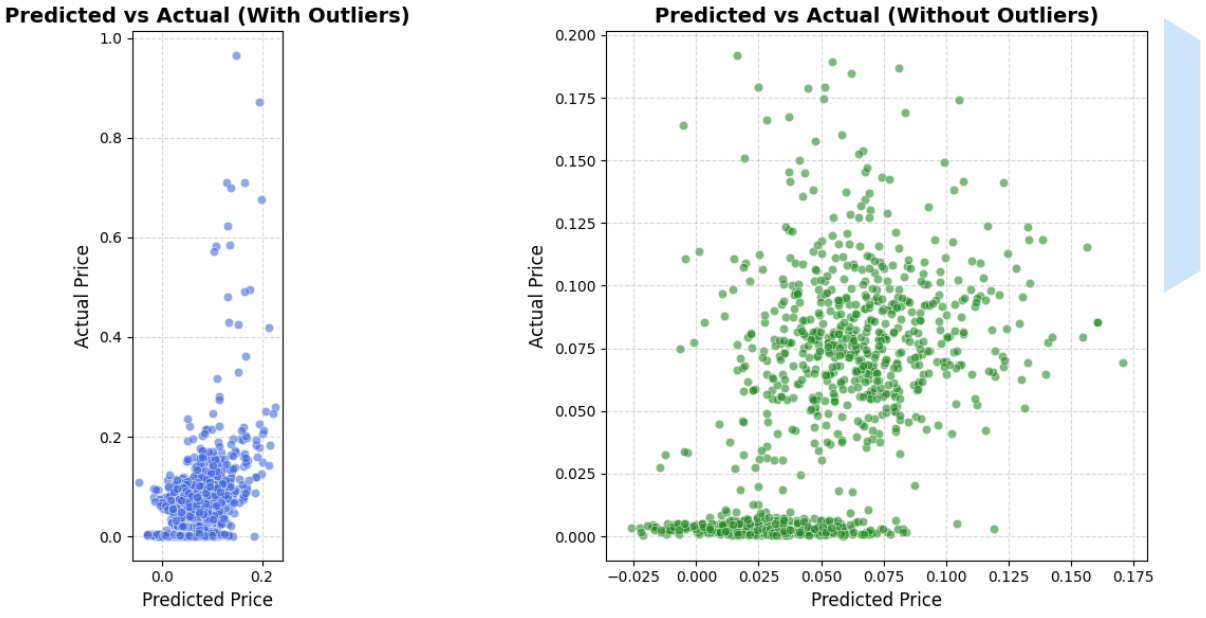
- لایه ورودی با ۱۳ نورون
- لایه پنهان با ۱۳ نورون
- لایه خروجی با ۱ نورون

و تمامی لایه ها به صورت تماما متصل میباشند. برای آموزش تمامی مدل ها در این گزارش کار از بهینه ساز SGD استفاده میکنیم و نرخ یادگیری را برابر با ۰.۰۰۱ قرار میدهیم(علت استفاده از این نرخ یادگیری آن است که به علت کوچک و نزدیک هم بودن فیلد هدف، باید از نرخ یادگیری کوچک استفاده شود تا مدل بتواند همگرا شود). همچنین تابع هزینه برابر با mse انتخاب شده، تعداد ایپاک برابر با ۱۵۰ و batchsize برابر با ۳۲ انتخاب شده است. لازم به ذکر است که در تمامی مدل ها، مقداری از داده های آموزش(۰.۱ آنها) به عنوان داده های راستی آزمایی جدا شده اند که خطای هر ایپاک از روی این دسته داده ها گزارش میشود. این پارامتر ها نیز از روی ازمون و خطا روی مدل اولیه شبکه به دست آمده اند.

در ابتدا، تمام توابع فعال سازی را خطی انتخاب میکنیم و شبکه را آموزش میدهیم. سپس نمودار خطا روی داده های آزمون را مشاهده کرده و میبینیم که هردو مدل به درستی آموزش داده شده و از خطای قابل قبولی برخوردارند:

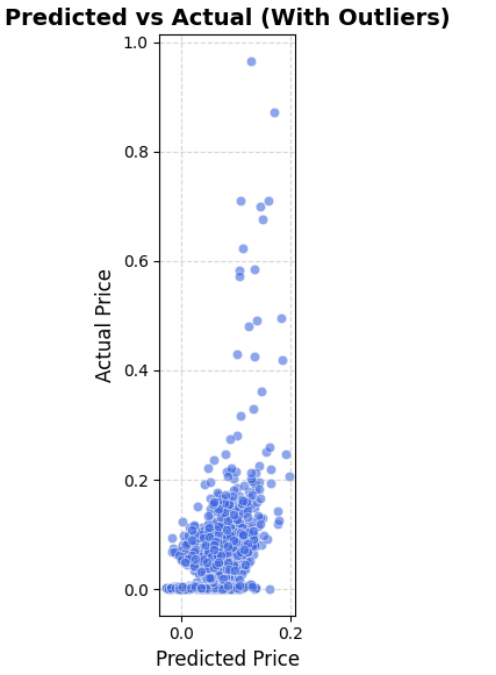


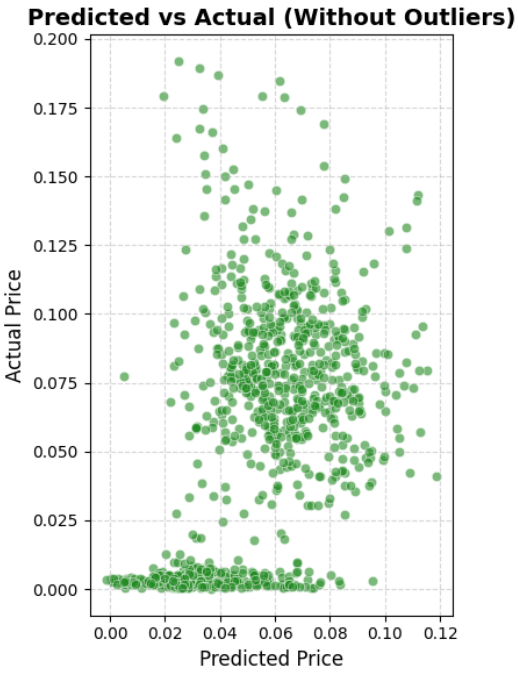
سپس مقادیر پیش بینی شده را با مقادیر واقعی مقایسه میکنیم:



همانطور که مشاهده میشود، در دیتای بدون داده های پرت، نتیجه نسبتا قابل قبولی به دست آمده، اما دیتای شامل داده های پرت، نتیجه جالبی ندارد. با حضور دیتا های پرت خطای mse برابر با 0.00509 و با حذف داده های پرت، خطا به ۰.۰۰۱۱۵ کاهش میابد.

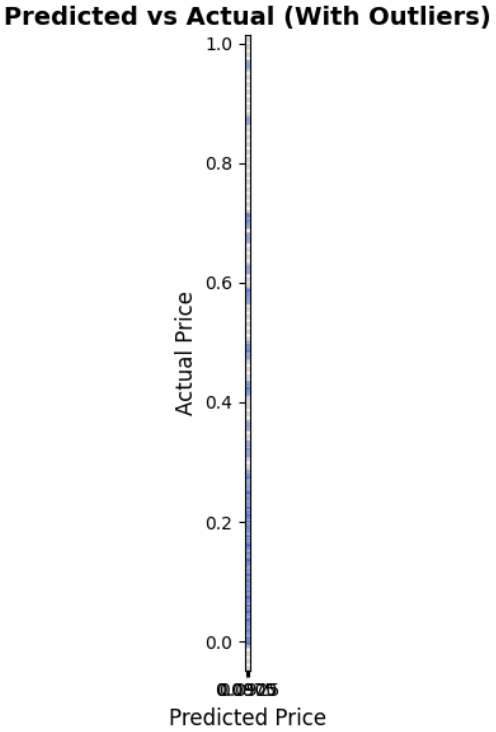
در مرحله بعد، تعداد لایه های شبکه را به ۶ افزایش میدهیم تا تاثیر افزایش لایه ها بر روی عملکرد مدل مشاهده شوند. در این شبکه نیز هر لایه(بجز لایه خروجی) ۱۳ نورون داشته و لایه خروجی نیز دارای ۱ نورون میباشد. در این شبکه، همچنان توابع فعال سازی خطی در نظر گرفته شده اند. برای هر دو مدل دیتا، یک شبکه مجزا را آموزش میدهیم و خروجی ها را مقایسه میکنیم:

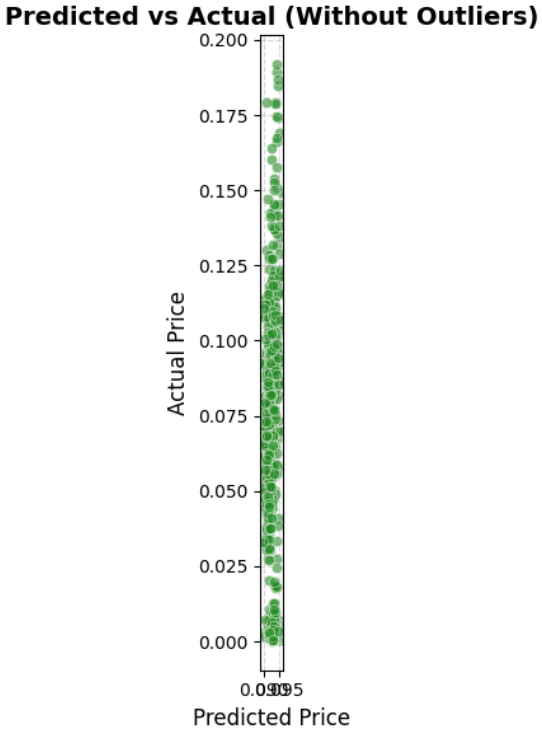




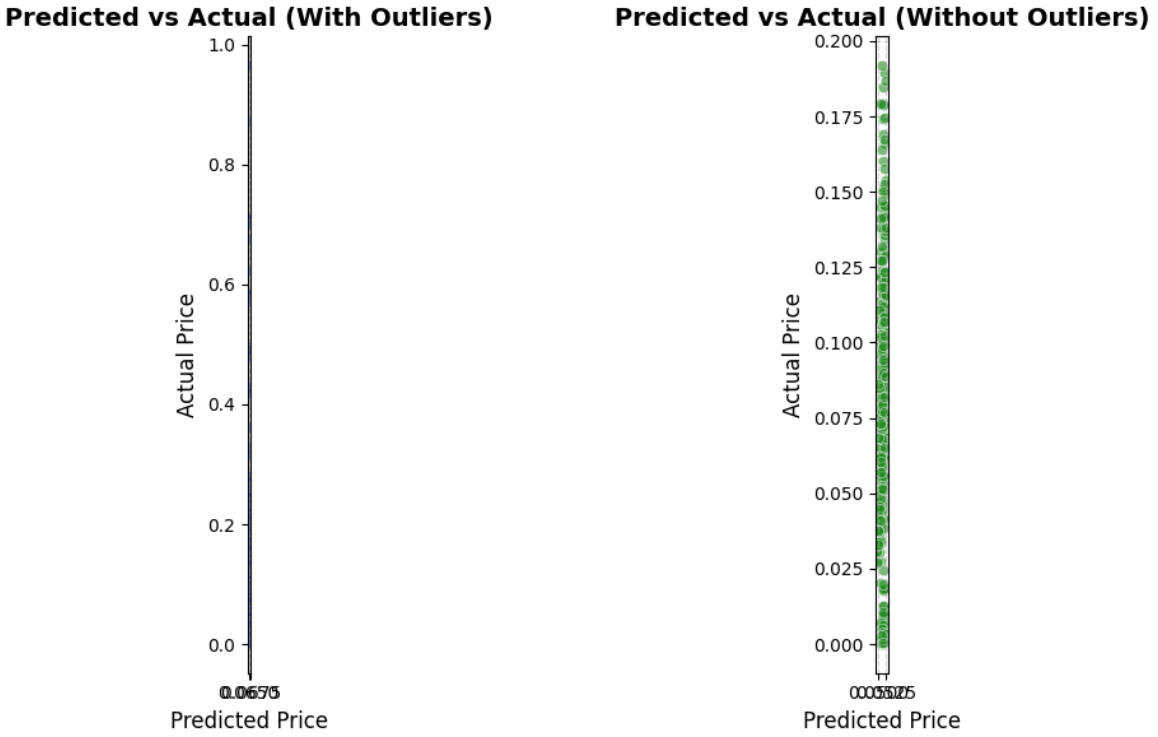
همانطور که مشاهده میشود، با افزایش تعداد لایه ها عملکرد شبکه اندکی بهتر میشود. در این حالت MSE برای دیتای شامل داده های پرت برابر با ۰.۰۰۵۳۱ و برای دیتای بدون داده های پرت برابر با ۰.۰۰۱۴۴ میشود.

در مرحله بعدی تلاش میکنیم تا با اضافه کردن توابع فعال سازی sigmoid، ویژگی های غیرخطی به شبکه خود آموزش بدهیم. برای این هدف، تابع فعال ساز ۳ لایه اخر هرکدام از شبکه های خود را از linear به sigmoid تغییر میدهیم و نتایج را مشاهده میکنیم:





همانطور که مشاهده میشود، این امر به بهتر شدن شبکه کمکی نکرد، چرا که بازه پیش بینی ها را کوچک کرده و از شبکه توانایی پیش بینی رنج گسترده داده ها را گرفت. سپس مدل دیگری با معماری ثابت اما با این تفاوت که ۳ لایه اول sigmoid و ۳ لایه بعدی خطی هستند را آموزش میدهیم، و مشاهده میکنیم که نتایج تفاوت چندانی ندارد:



بنابراین بهترین مدل به دست آمده از این مرحله، مدل ۶ لایه تمام متصل با توابع فعالسازی خطی میباشد. برای بهتر کردن مدل در مراحل آینده، از همین شبکه به دست آمده استفاده شده و تلاش بر بهبود آن میشود.

در مسائل رگرسیون، هدف پیش‌بینی مقادیر پیوسته است، بنابراین انتخاب تابع فعال‌سازی تأثیر زیادی بر عملکرد مدل دارد. هنگامی که از تابع فعال‌سازی سیگموید یا ReLU در لایه‌های پنهان استفاده می‌شود، ممکن است برخی از مشکلات مانند اشباع شدن گرادیان یا تغییر مقیاس نامناسب داده‌ها رخ دهد. سیگموید خروجی را بین صفر و یک محدود می‌کند که در مسائل رگرسیون می‌تواند باعث فشرده شدن مقادیر و از دست رفتن اطلاعات شود. همچنین، ReLUبرای مقادیر منفی خروجی صفر تولید می‌کند که ممکن است

بر روند یادگیری تأثیر منفی بگذارد. در مقابل، استفاده از تابع خطی در تمام لایه‌ها باعث می‌شود که مدل بدون اعمال تغییرات غیرضروری بر مقادیر، الگوهای داده را به‌درستی یاد بگیرد.

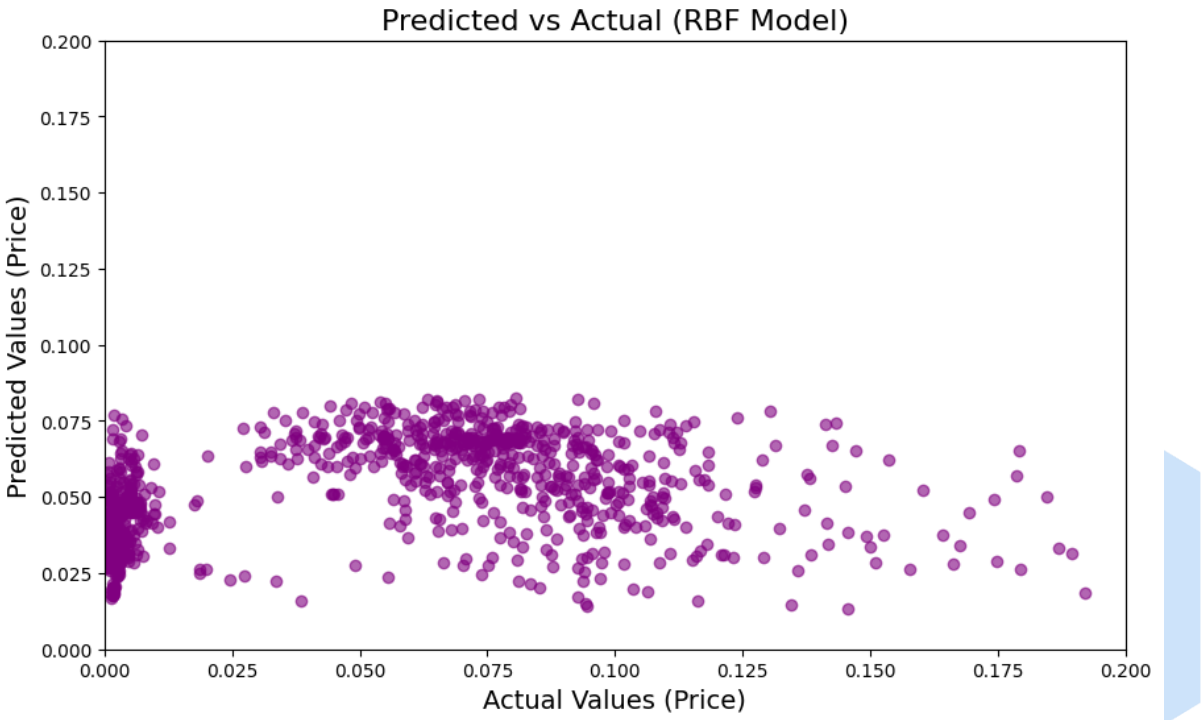
وقتی تمام توابع فعال‌سازی در شبکه MLP خطی باشند، مدل اساساً یک ترکیب خطی از ورودی‌ها را یاد می‌گیرد که در بسیاری از مسائل رگرسیون می‌تواند عملکرد بهتری داشته باشد، مخصوصاً اگر رابطه بین متغیرهای ورودی و خروجی ذاتاً خطی یا تقریباً خطی باشد. استفاده از توابع غیرخطی زمانی ضروری است که داده‌ها دارای روابط پیچیده و غیرخطی باشند. اما در صورتی که رابطه داده‌ها تقریباً خطی باشد، توابع غیرخطی ممکن است یادگیری را پیچیده کرده و به بیش‌برازش یا کاهش دقت منجر شوند. بنابراین، اگر مدل شما با توابع خطی عملکرد بهتری داشته، به این معناست که داده‌ها ویژگی‌های غیرخطی پیچیده‌ای ندارند و یک ترکیب خطی از متغیرهای ورودی برای پیش‌بینی مقادیر کافی بوده است.

اضافه کردن لایه RBF به شبکه به دست آمده

اضافه کردن یک لایه RBFبه ابتدای شبکه عصبی باعث می‌شود که داده‌های ورودی به یک فضای جدید منتقل شوند که در آن روابط غیرخطی بهتر نمایش داده شوند. این لایه با استفاده از توابعی مانند گاوسی، ویژگی‌های جدیدی ایجاد می‌کند که تفکیک‌پذیری داده‌ها را افزایش می‌دهند. در نتیجه، شبکه می‌تواند الگوهای پیچیده را راحت‌تر تشخیص دهد و عملکرد بهتری در مسائل رگرسیون داشته باشد. علاوه بر این، چون توابع RBF فقط بر اساس فاصله از مراکز خود مقداردهی می‌شوند، مدل می‌تواند روی الگوهای محلی داده تمرکز کند و از تأثیر منفی نویز یا مقادیر پرت کاسته شود.

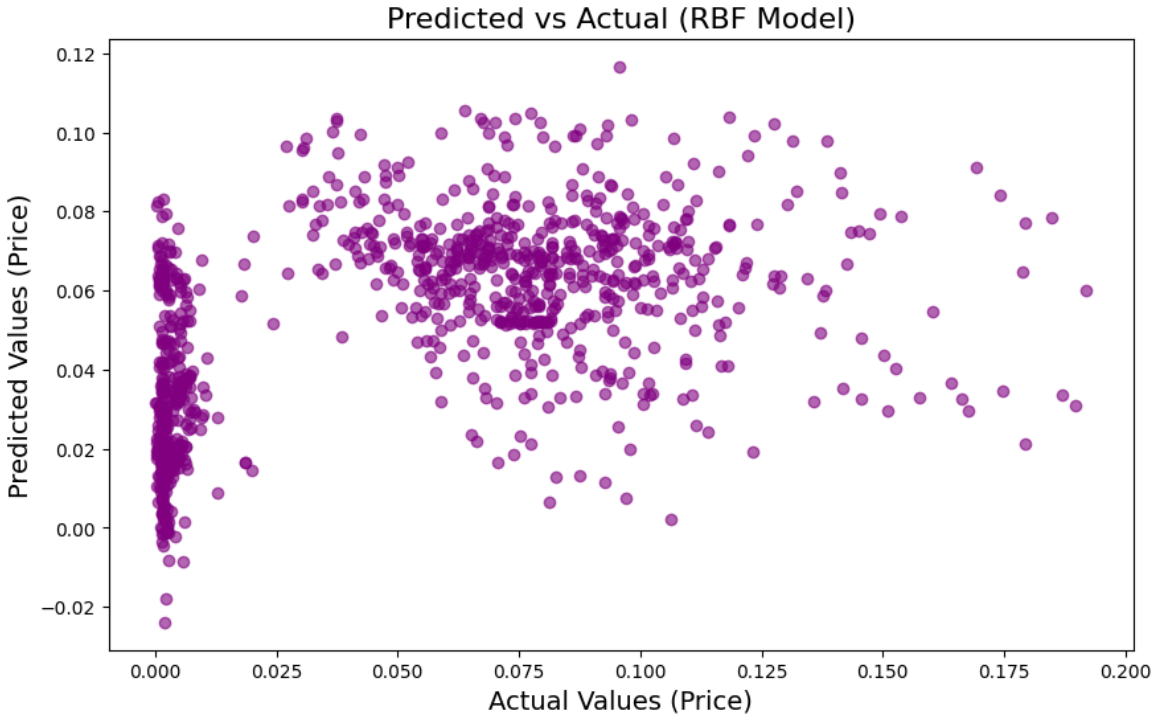
ما در این بخش، لایه ورودی را تبدیل به یک لایه RBF میکنیم و با تغییر تعداد مراکز، تاثیر آن روی نتیجه را مشاهده میکنیم. برای این کار، از انجا که tensorflow تابع آماده ای برای RBF ندارد، یک کلاس RBF مینویسیم. این کلاس یک لایه RBFبرای شبکه عصبی تعریف می‌کند که داده‌های ورودی را به یک فضای غیرخطی تبدیل می‌کند. ابتدا مراکز RBFو عرض‌های آن‌ها به‌عنوان پارامترهای قابل آموزش مقداردهی اولیه می‌شوند. سپس هنگام پردازش ورودی، فاصله هر نمونه تا این مراکز محاسبه شده و با استفاده از تابع گاوسی تبدیل می‌شود. این تبدیل باعث می‌شود ویژگی‌های ورودی در یک فضای جدید نمایش داده شوند که به شبکه کمک می‌کند الگوهای پیچیده را بهتر یاد بگیرد.

ابتدا ۱۰ مرکز را برای شبکه RBF خود انتخاب میکنیم و آن را به بهترین شبکه به دست آمده از مرحله قبل، یعنی شبکه ۶ لایه با تمام توابع فعال سازی برابر با linear اعمال میکنیم. همچنین همانطور که از مرحله قبل متوجه شدیم، دیتایی که از آن داده های پرت حذف شده اند نتیجه بهتری نسبت به دیتایی دارد که در آن داده های پرت وجود دارند، پس ادامه کار را با دیتایی جلو میبریم که در آن داده های پرت حذف شده اند. در این حالت داریم:



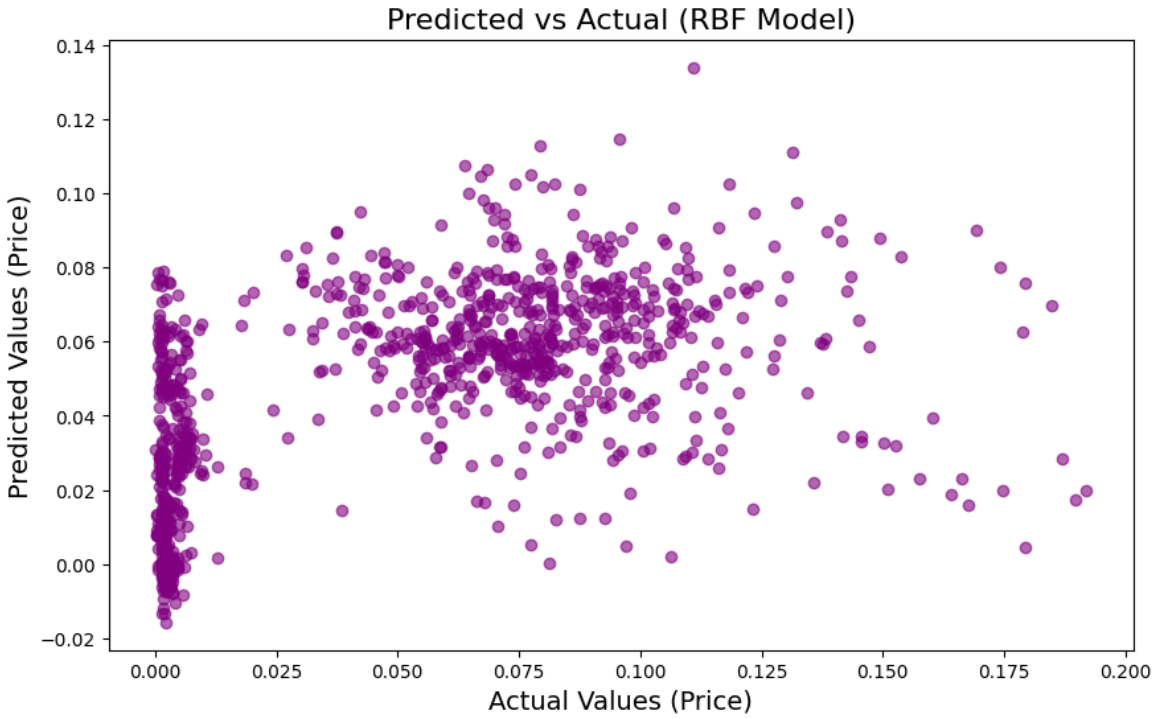
که در آن mse برابر با ۰.۰۰۱۷۵ میباشد. همانطور که مشاهده میشود، پیش بینی اندکی بهتر شده اما همچنان قابل قبول نمیباشد.

در مرحله بعد، تعداد مراکز را به ۱۰۰ افزایش میدهیم و نتایج را مشاهده میکنیم:

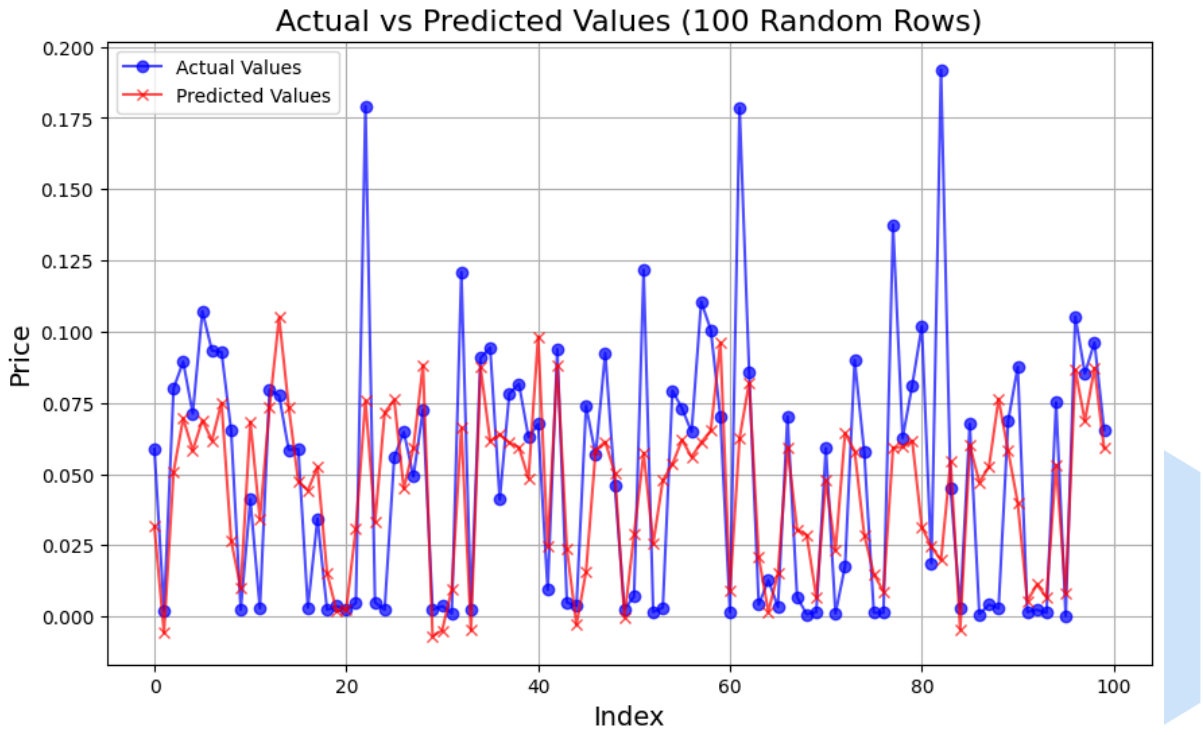


در این حالت، نتایج برای داده ها بهتر میشود اما همچنان تعداد زیادی از داده ها به صورت غلط تخمین زده میشوند، در این حالت mse برابر با ۰.۰۰۱۴۴ به دست می آید.

در مرحله اخر نیز تعداد مراکز را به ۸۰۰ افزایش میدهیم و نتایج را مشاهده میکنیم:



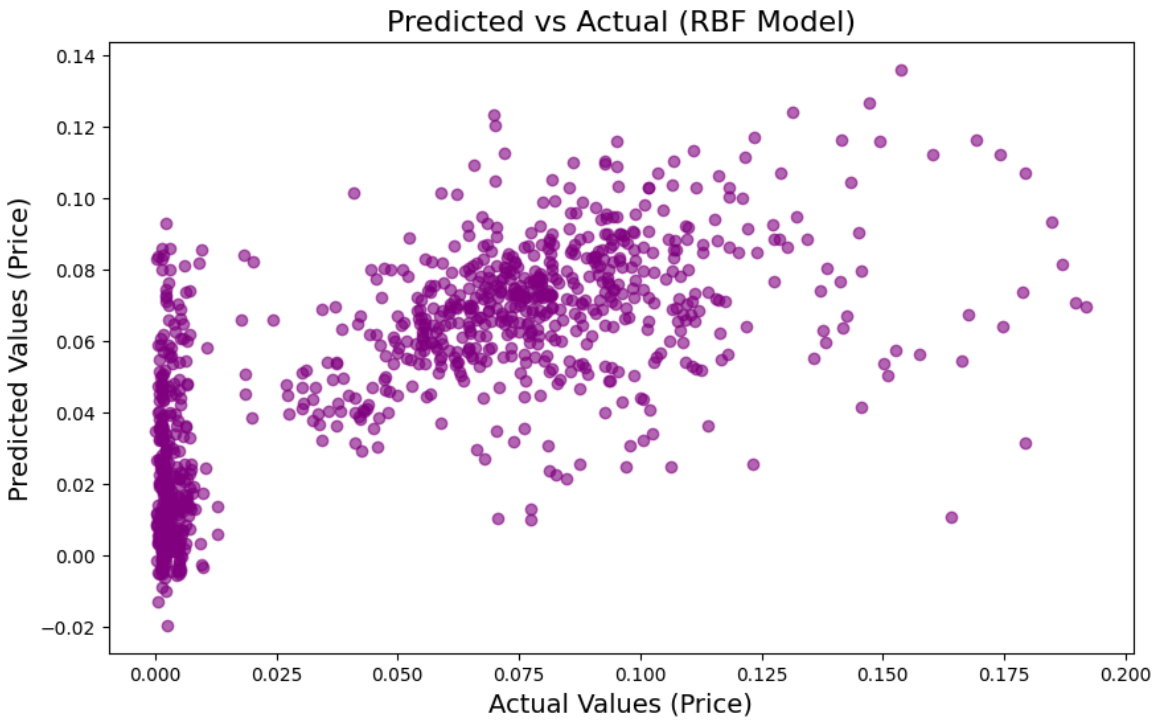
در این حالت نیز نتایج نسبت به حالت قبلی بهبود پیدا میکنند اما همچنان نتایج قابل قبولی نیستند، در این حالت mse برابر با ۰.۰۰۱۳۸ به دست آمده و همچنین r2-score آن برابر با ۰.۳ به دست می آید که هرچند از نمونه های قبلی بهتر است، اما همچنان دقت کافی را برای انجام یک تسک رگرشن ندارد. که نشاندهنده آن است که اگرچه مدل خطای عمومی کمی دارد، اما در پیدا کردن الگوی نهان در داده ها خوب عمل نمیکند. برای مشاهده بهتر این امر، تخمین چند داده رندوم را مشاهده میکنیم:



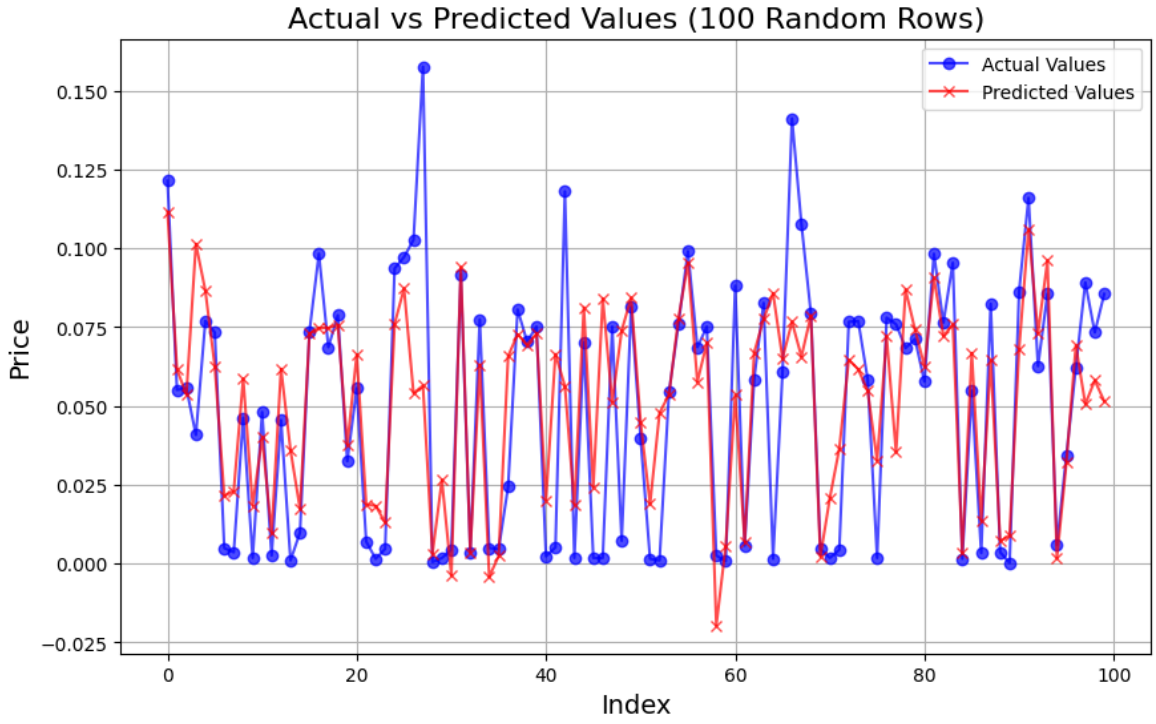
همانطور که مشاهده میشود، مدل در تخمین داده هایی که روی Q2 هستند به صورت قابل قبولی عمل میکند، اما هرچه انحراف داده ها از میانه بیشتر میشود، خطای مدل نیز افزایش میابد.

حال که به نظر میرسد مدل mlp توانایی عملکرد بهتری را ندارد، شروع به عوض کردن پارامتر های مدل میکنیم. در مرحله اول، برای مدل با ۴ لایه پنهانی که داشتیم(هر لایه پنهان دارای ۱۳ نورون)، تعداد نورون های هر لایه پنهان را به ۵۰ افزایش میدهیم و همانطور که در ابتدا ذکر شد، solver برای تمامی مراحل sgd در نظر گرفته شده بود، با SGD solver، با این تغییر R2 score عددی منفی میشد، اما وقتی solver را به Adam تغییر دادیم، مشاهده شد که R2 score به ۰.۵۲ میرسد.

در این حالت، شکل خروجی ها به شکل زیر تبدیل میشود:



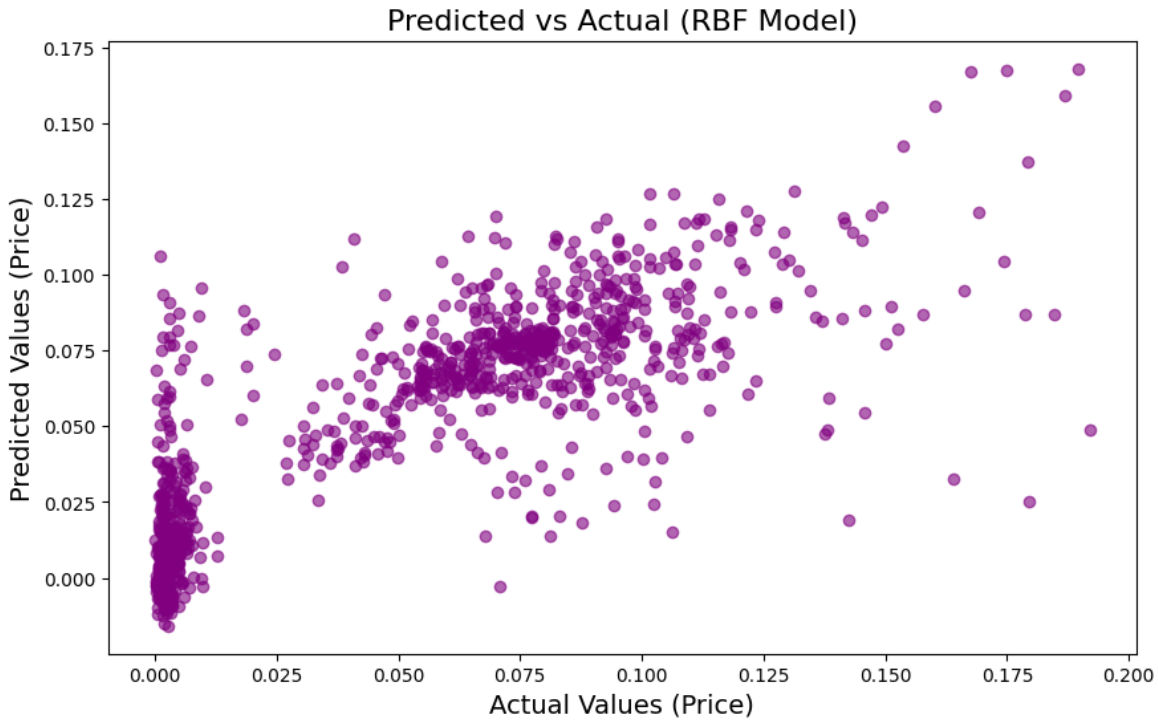
و اگر بخواهیم عملکرد مدل را روی ۱۰۰ داده رندوم مشاهده کنیم، تخمین مدل به صورت زیر خواهد بود که مشاهده میشود نسبت به حالت قبل، پیشرفت چشم گیری داشته است:



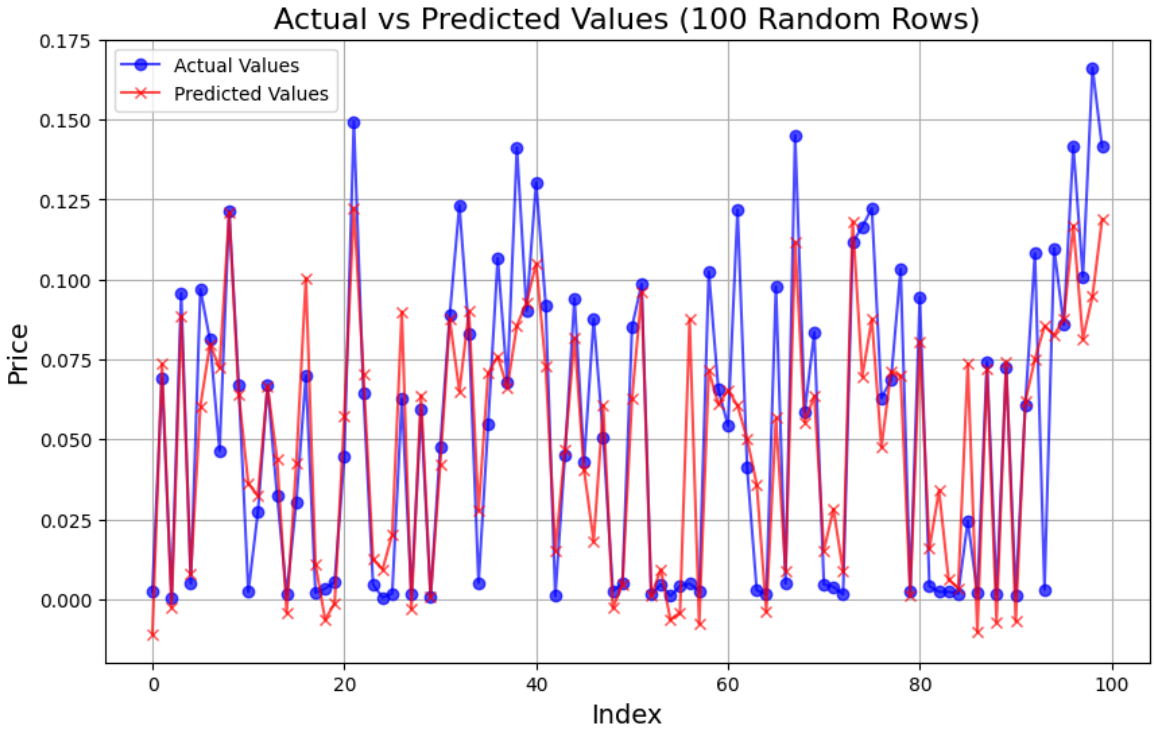
پس از تست کردن چندین مدل مختلف، بهترین نتیجه برای شبکه هایی با لایه های پنهان ۱۰۰*۲۰۰*۲۰۰*۱۰۰ و ۱۰۰*۲۰۰*۱۰۰ و هردو مدل R2 score برابر با ۰.۵۶ و MSE برابر با ۰.۰۰۰۸۵ را دارا هستند.

حال که متوجه شدیم با تعویض solver به مقادیر بهتری میتوانیم دست پیدا کنیم، این نتیجه را با درست کردن Pipeline و اضافه کردن RBF به ابتدای شبکه ۴ لایه ای به دست آمده از مرحله قبل دوباره تست میکنیم. با ۱۰ سنتر، R2 برابر با ۰.۴۲، با ۱۰۰ سنتر برابر با ۰.۵۲ و با ۸۰۰ سنتر (مانند بهترین نتیجه قسمت قبل) R2 برابر با ۰.۶۵ به دست آمده و MSE برابر با ۰.۰۰۰۶۸ میشود.

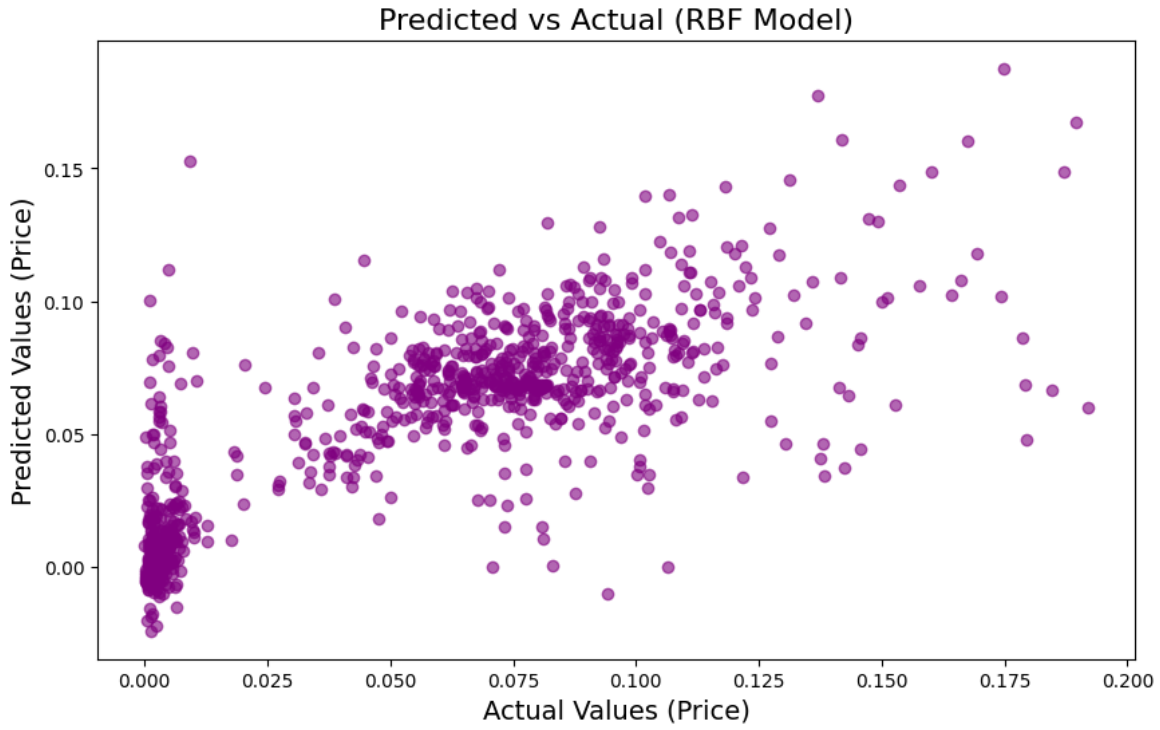
شکل پیش بینی های مدل با۸۰۰ سنتر:



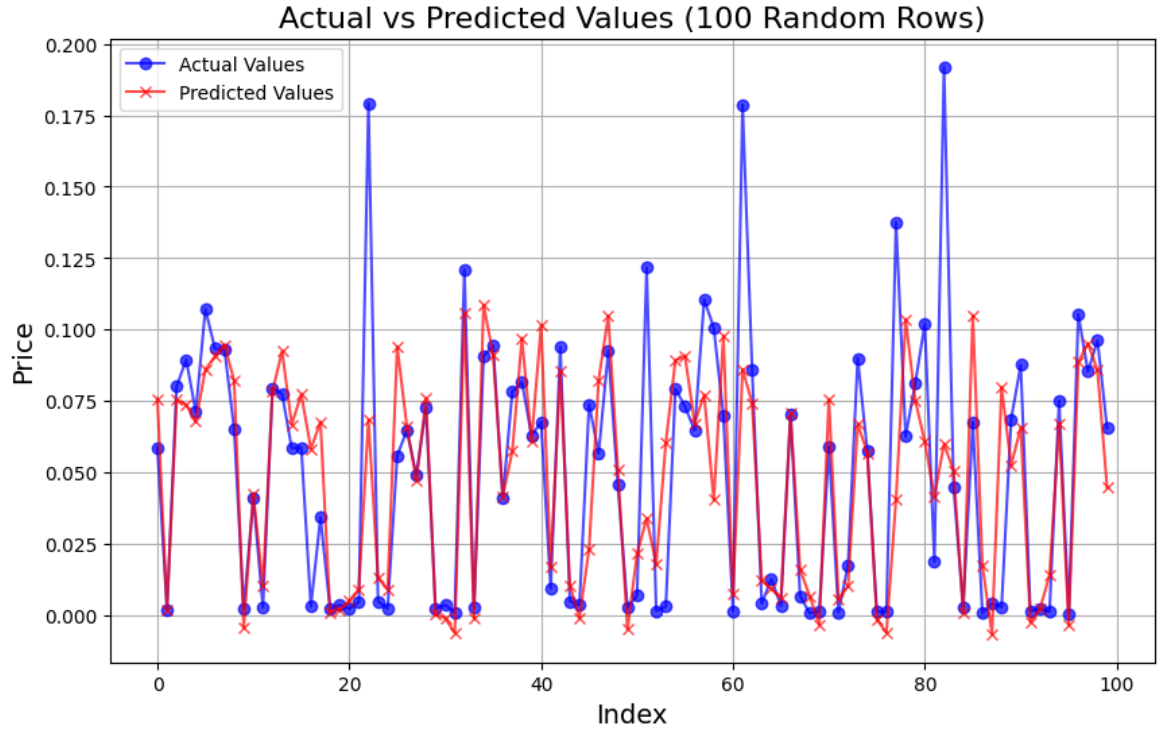
پیش بینی های مدل روی ۱۰۰ داده رندوم:



سپس برای بهتر کردن عملکرد مدل، در ابتدای pipeline از feature engineering استفاده میکنیم، با این کار از دستور polynomial_features استفاده میکنیم، زمانی که از PolynomialFeatures با درجه ۲ استفاده می‌شود، علاوه بر ویژگی‌های اصلی، ترکیبات دوتایی بین آن‌ها به عنوان ویژگی‌های جدید تولید می‌شود. این فرایند باعث می‌شود که مدل بتواند روابط غیرخطی میان ویژگی‌ها را بهتر یاد بگیرد زیرا تعاملات میان آن‌ها به صورت یک ویژگی خطی نمایش داده می‌شود. در نتیجه، افزودن ویژگی‌های چندجمله‌ای درجه دوم به بهبود توانایی مدل در شناسایی الگوهای پیچیده و تعاملات میان متغیرها کمک می‌کند. پس از اضافه کردن این دستور به ابتدای pipeline، نتیجه به صورت زیر میشود:



و پیش بینی مدل برای ۱۰۰ داده رندوم نیز به صورت زیر تبدیل میشود:

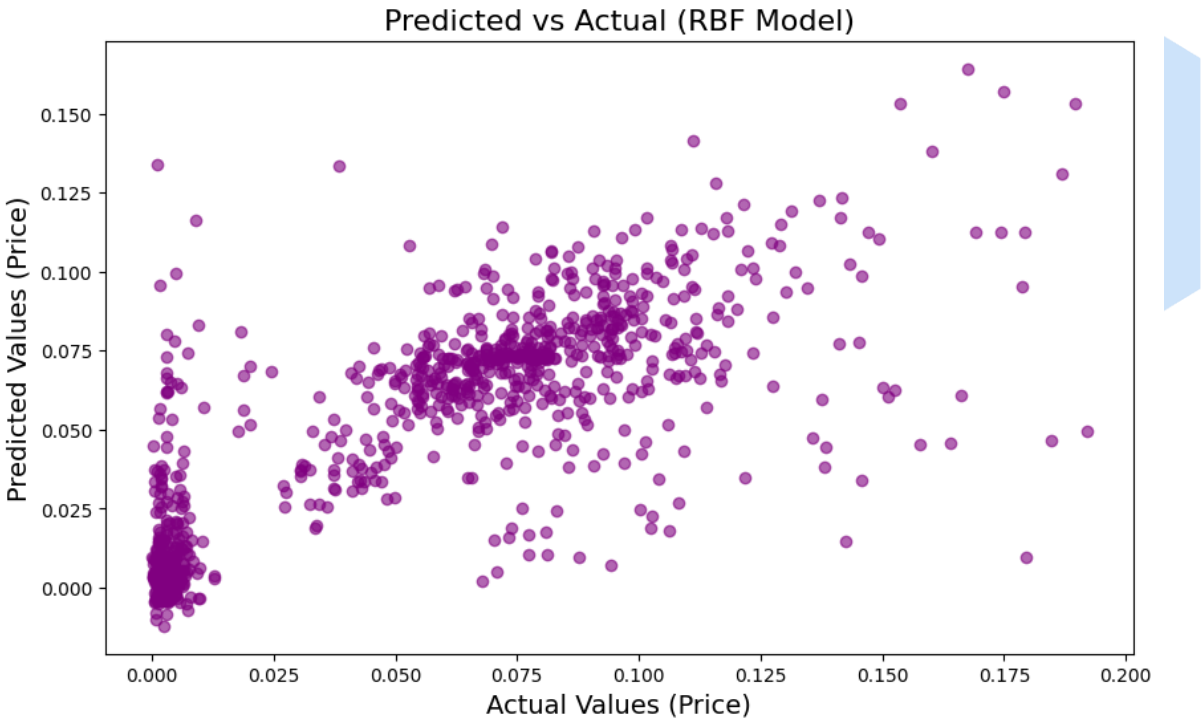


با انجام این کار R2 به ۰.۶۶ افزایش یافته و mse برابر با ۰.۰۰۰۶۶ میشود.

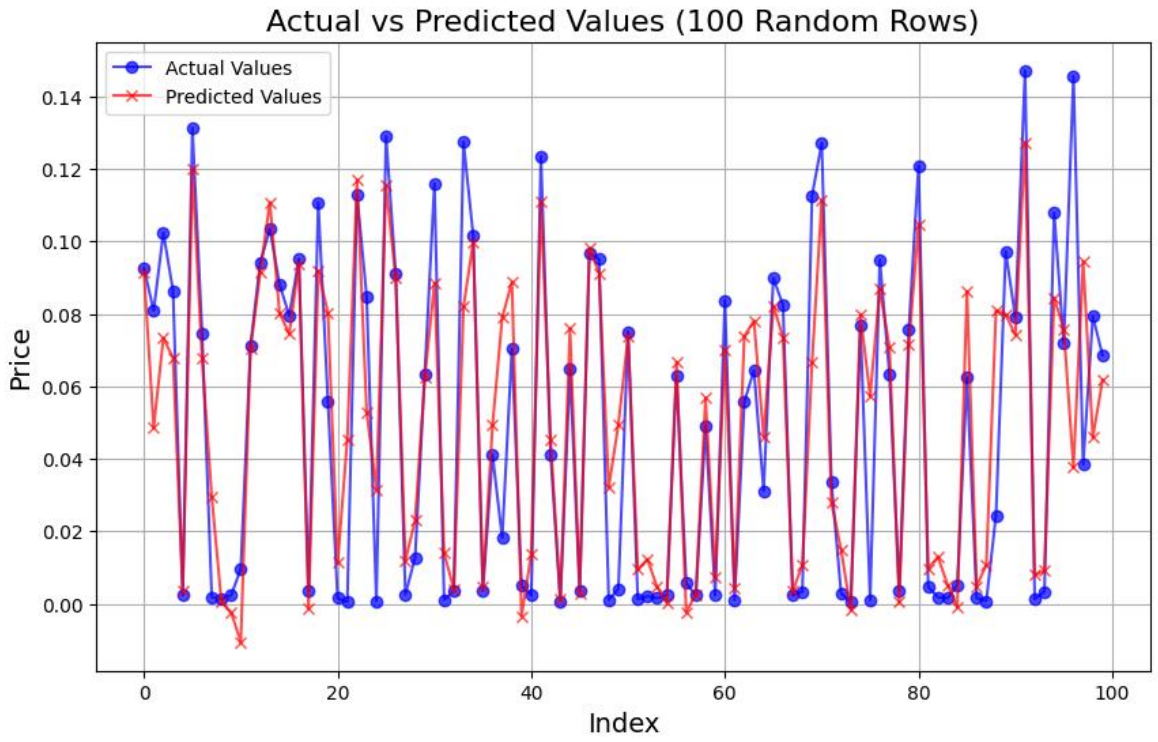
در مرحله بعد، برای آن که مطمین شویم بهترین R2 ممکن از بین تمامی مدل ها به دست آمده، و همچنین برای مشاهده تاثیر حذف برخی فیچر ها، از grid search استفاده میکنیم تا مجبور نشویم تمامی حالات ممکن را دستی تست کنیم. در این grid search، حالات زیر تست میشوند:

- فیچر های مرتبه ۱ و ۲ و ۳
- انتخاب بهترین فیچر ها و حذف باقی آنها(انتخاب ۵، ۱۰، ۲۰ فیچر)
- انتخاب بهترین مدل شبکه(100*200*100و 100*200*200*100)
- انتخاب بهترین تعداد سنتر ها(۱۰۰، ۸۰۰ و ۱۲۰۰)
- انتخاب بهترین نرخ یادگیری(۰.۰۱ یا ۰.۰۰۱)
- انتخاب بهترین آلفا برای جلوگیری از اورفیتینگ(۰.۰۰۱ یا ۰.۰۰۰۱)

در این حالت، R2 score برابر با ۰.۶۷ به دست آمده و MSE نیز برابر با ۰.۰۰۰۶ به دست می آید. در این حالت خروجی تمام داده های تست به صورت:



و خروجی مدل روی ۱۰۰ داده رندوم به این صورت به دست می آید:



همانطور که مشاهده میشود، این بهترین مدل به دست آمده از مجموع تمامی کارهای انجام شده این مدل میباشد:

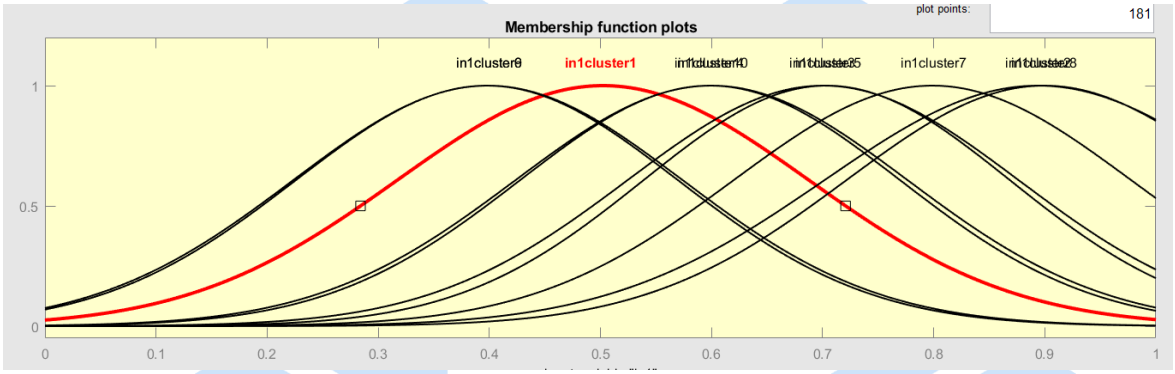
مشخصات مدل:

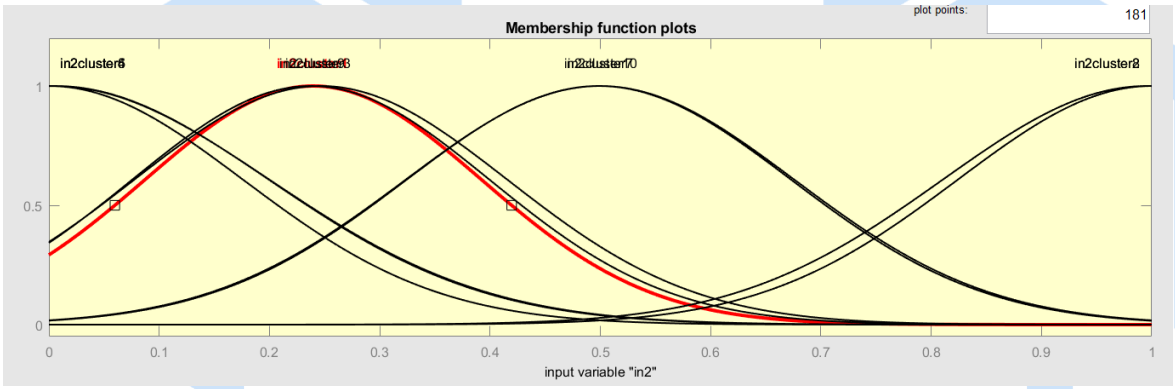
{'mlp__alpha': 0.001, 'mlp__hidden_layer_sizes': (100, 200, 100), 'mlp__learning_rate_init': 0.001, 'poly__degree': 1, 'rbf__n_components': 800, 'select__k': 20}

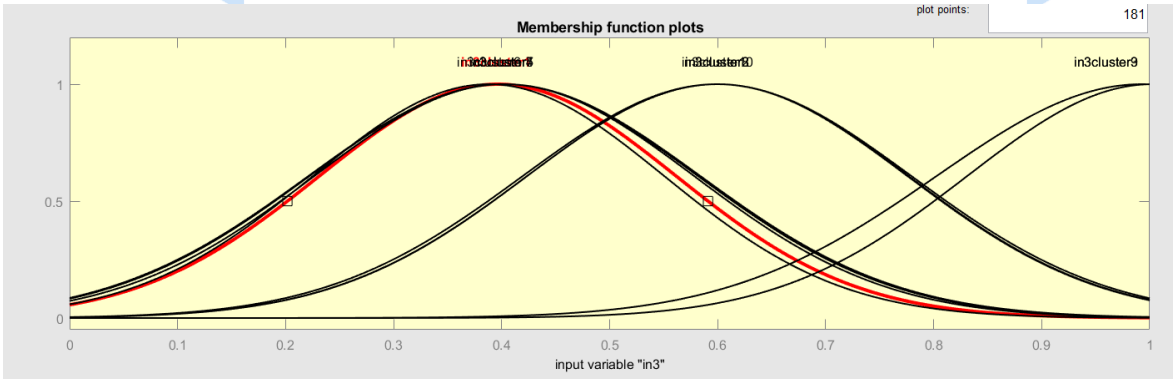
استفاده از مدل فازی

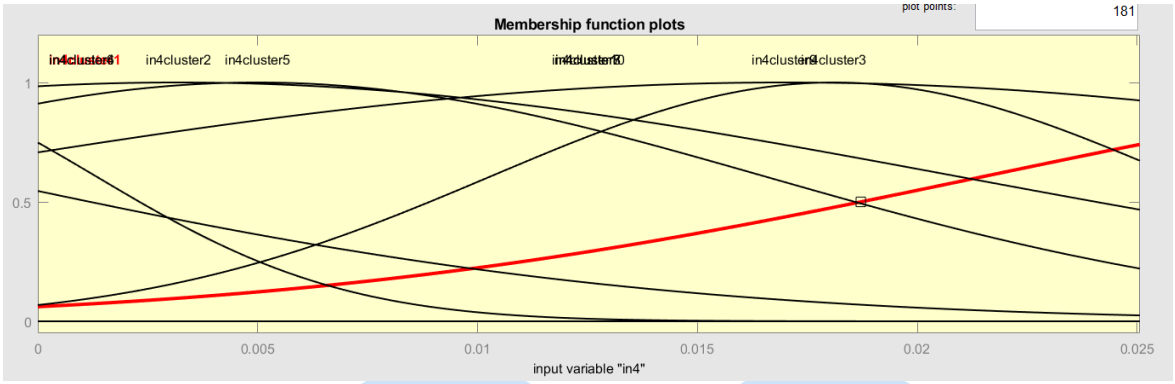
سپس برای اطمینان از بهترین جواب ممکن بودن این تست، دیتای trim را به محیط متلب انتقال میدهیم و ابتدا یک مدل FIS ابتدایی به کمک subtractiveClustring روی آن ایجاد میکنیم و سپس مدل را به کمک دستور ANFIS با ۱۰۰ ایپاک آموزش میدهیم.

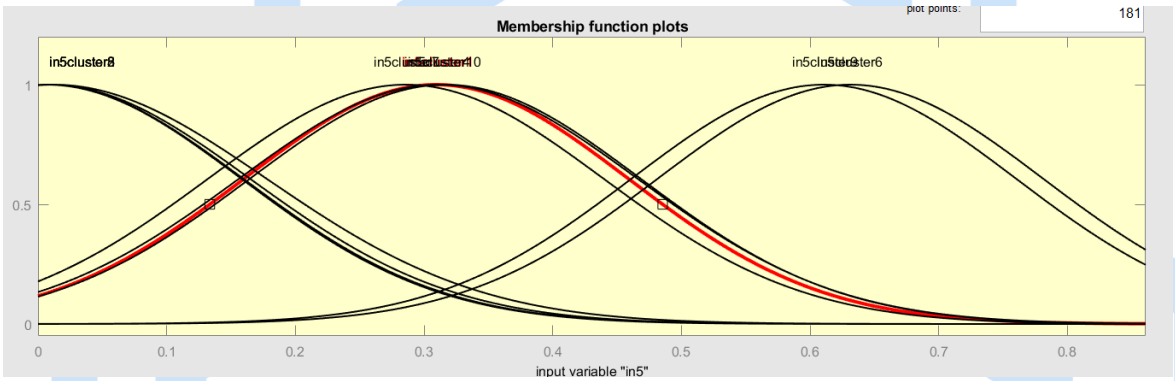
شکل توابع تعلق برای متغیر های ۱ تا ۱۳:

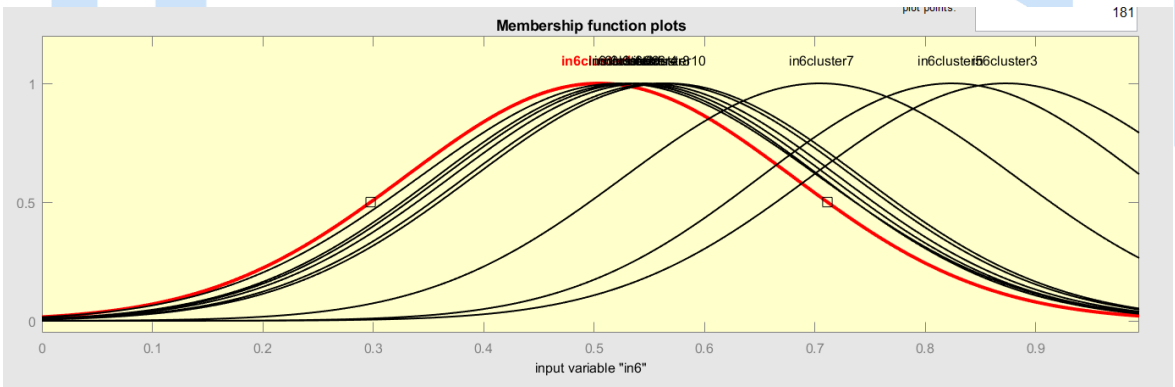


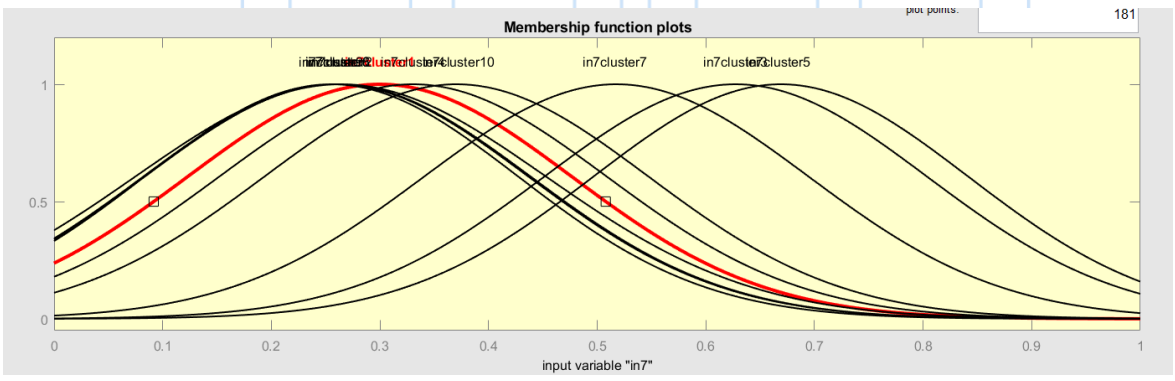


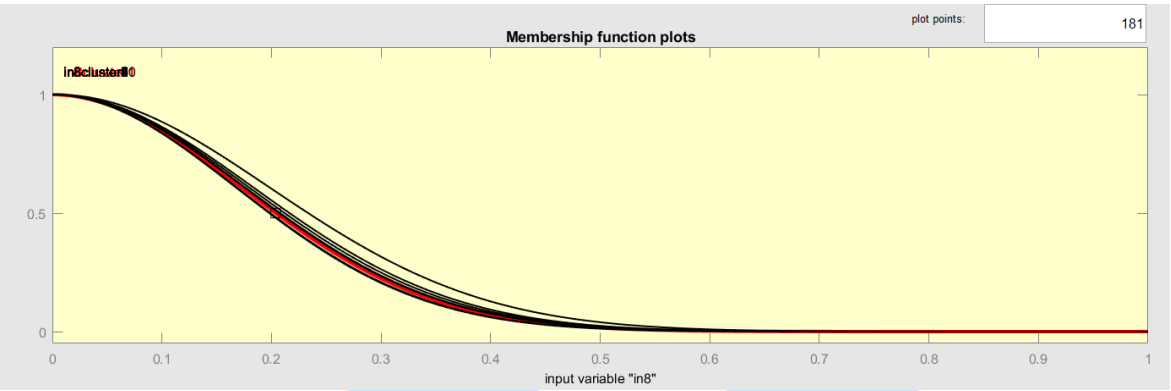


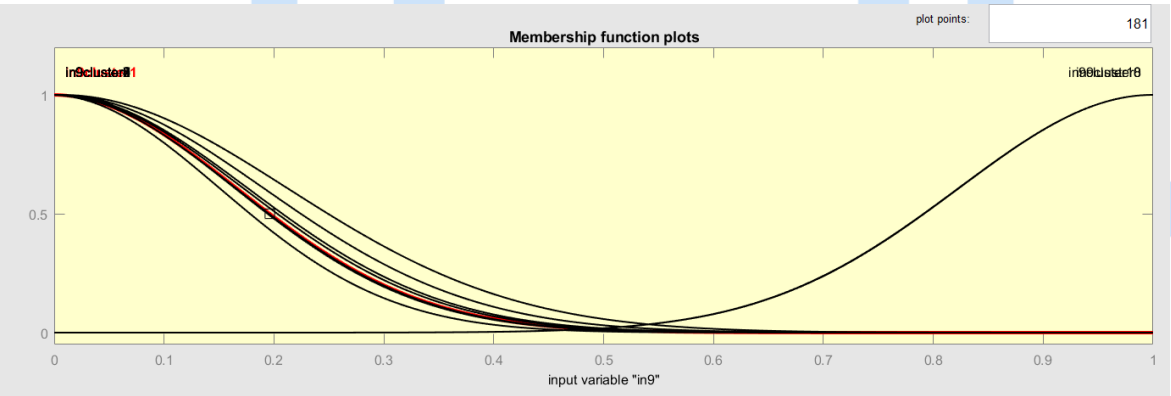

Membership function plots
plot points:
181
in4cluster2
in4cluster5
in4cluster3
1
0.5
0
0
0.005
0.01
0.015
0.02
0.025
input variable "in4"

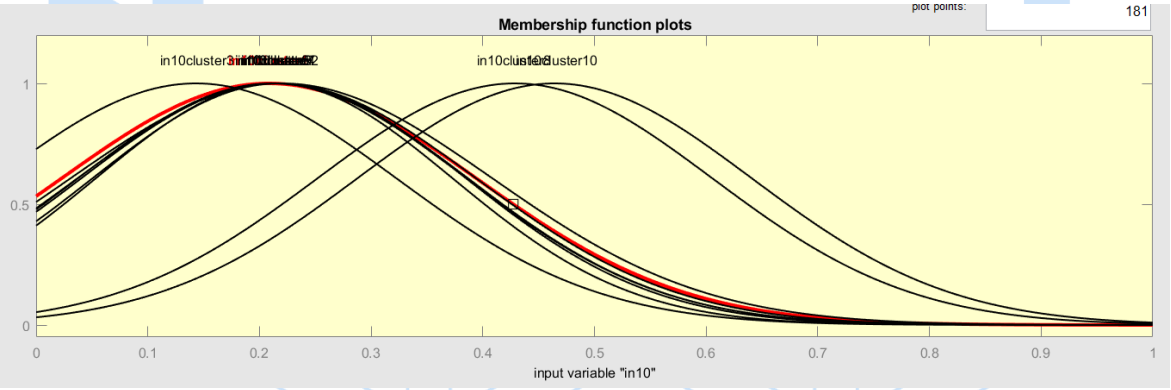

Membership function plots
plot points:
181
in5cluster8
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
input variable "in5"

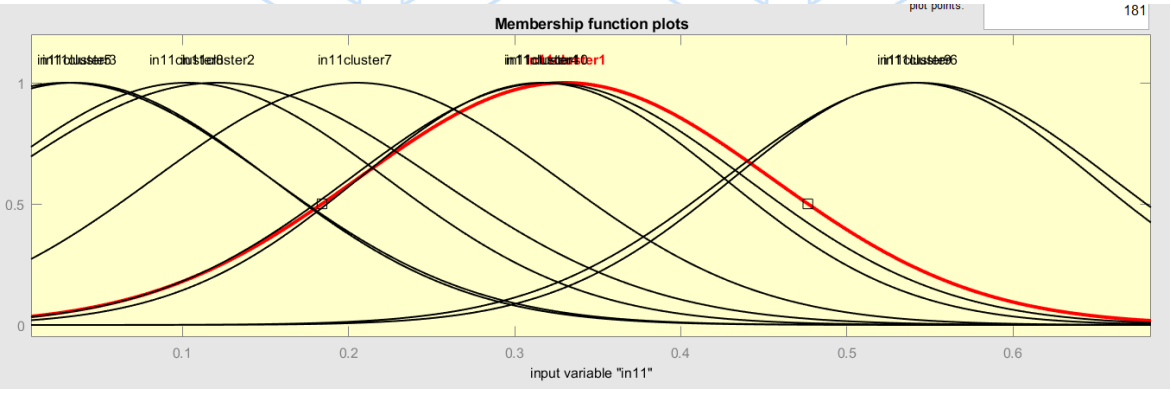

Membership function plots
plot points:
181
in6cluster7
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
input variable "in6"

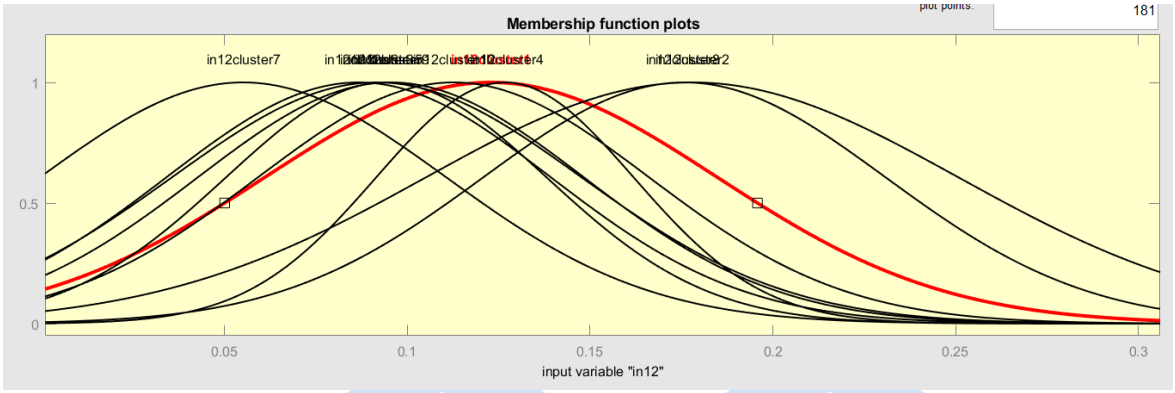

Membership function plots
plot points:
181
in7cluster7
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
input variable "in7"

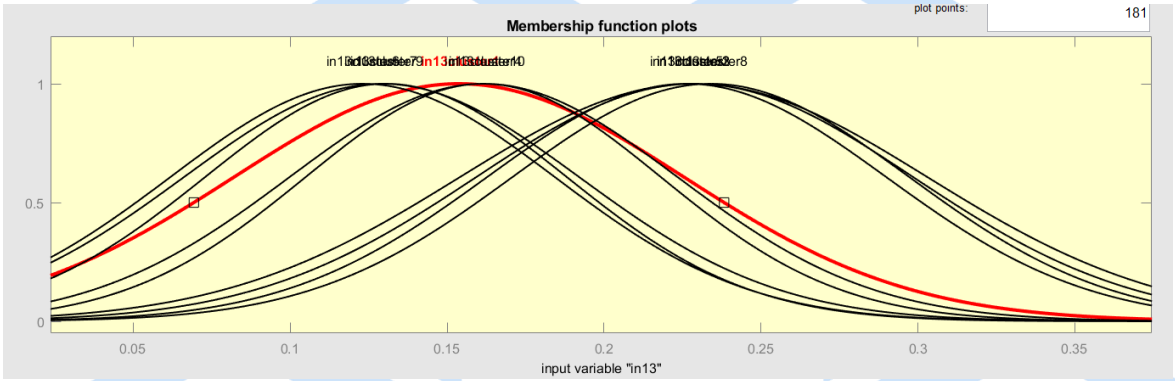

plot points:
181
Membership function plots
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
input variable "in8"


plot points:
181
Membership function plots
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
input variable "in9"

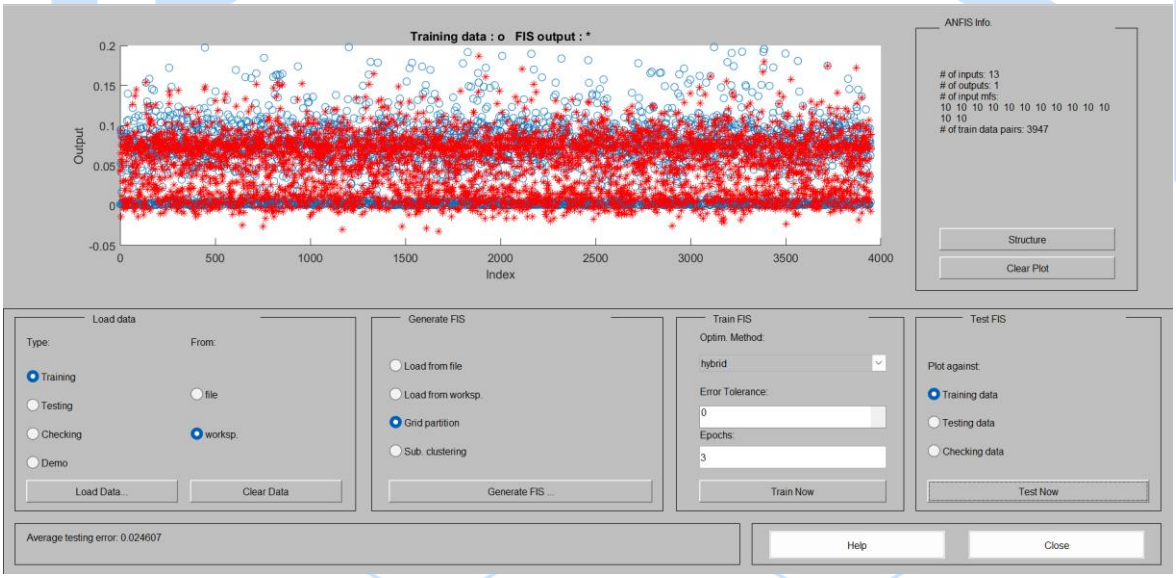

plot points:
181
Membership function plots
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
input variable "in10"


plot points:
181
Membership function plots
in11cluster2
in11cluster7
1
0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
input variable "in11"

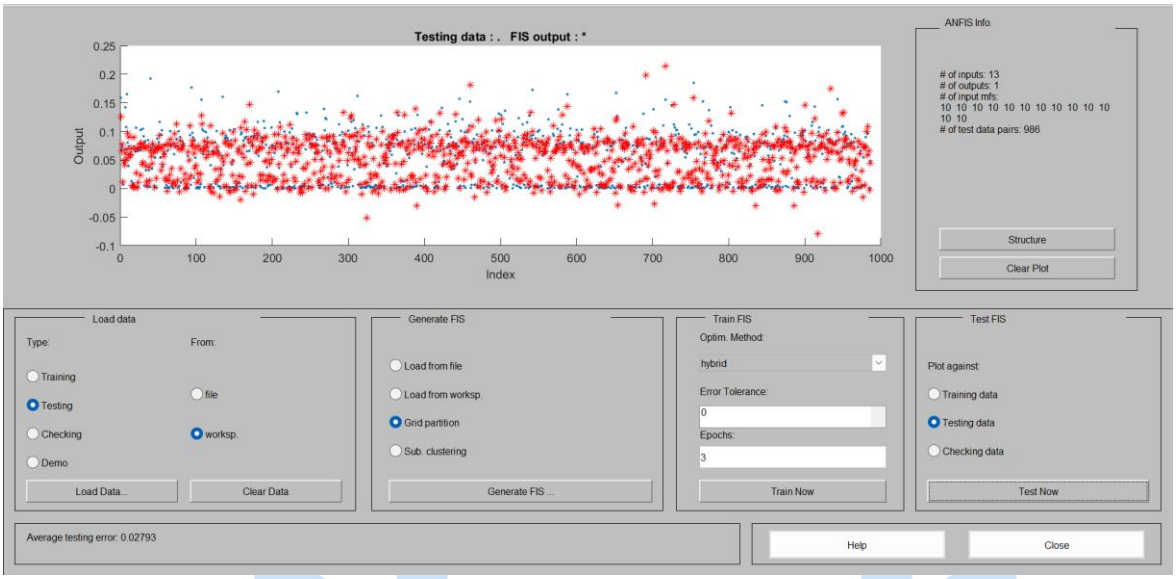




خروجی مدل فازی روی داده های ترین:



خروجی مدل فازی روی داده های تست:



در نهایت، MSE روی داده های تست برابر با 0.0078 و R2 برابر با 0.594 به دست آمد.

بخش۲: تخمین نوع با استفاده از SVM

ماشین بردار پشتیبان (SVM) یک الگوریتم یادگیری نظارت‌شده است که برای طبقه‌بندی و رگرسیون استفاده می‌شود. در حالت کلی، SVMیک ابرصفحه بهینه در فضای ویژگی‌ها پیدا می‌کند که داده‌های دو کلاس را از هم جدا می‌کند. این ابرصفحه به گونه‌ای انتخاب می‌شود که فاصله (margin) بین نزدیک‌ترین نقاط هر کلاس (که بردارهای پشتیبان نامیده می‌شوند) حداکثر باشد. در صورتی که داده‌ها خطی جداپذیر نباشند، SVMمی‌تواند با استفاده از تابع کرنل آن‌ها را به فضای با ابعاد بالاتر نگاشت کند تا جداپذیری بهبود یابد.

در روش یک در برابر همه(One-vs-All) که در این سوال استفاده شده، برای طبقه‌بندی سه‌کلاسه، سه مدل SVM جداگانه آموزش داده می‌شود. هر مدل یکی از کلاس‌ها را به‌عنوان کلاس مثبت و دو کلاس دیگر را به‌عنوان کلاس منفی در نظر می‌گیرد. در مرحله پیش‌بینی، هر نمونه جدید توسط هر سه مدل ارزیابی می‌شود و هر مدل یک مقدار امتیاز بازگشت می‌دهد. در نهایت، مدلی که بیشترین مقدار تصمیم‌گیری را ارائه دهد، تعیین‌کننده‌ی کلاس نهایی نمونه خواهد بود. این روش مقیاس‌پذیر است اما ممکن است در داده‌هایی که همپوشانی دارند، چالش‌هایی ایجاد کند.

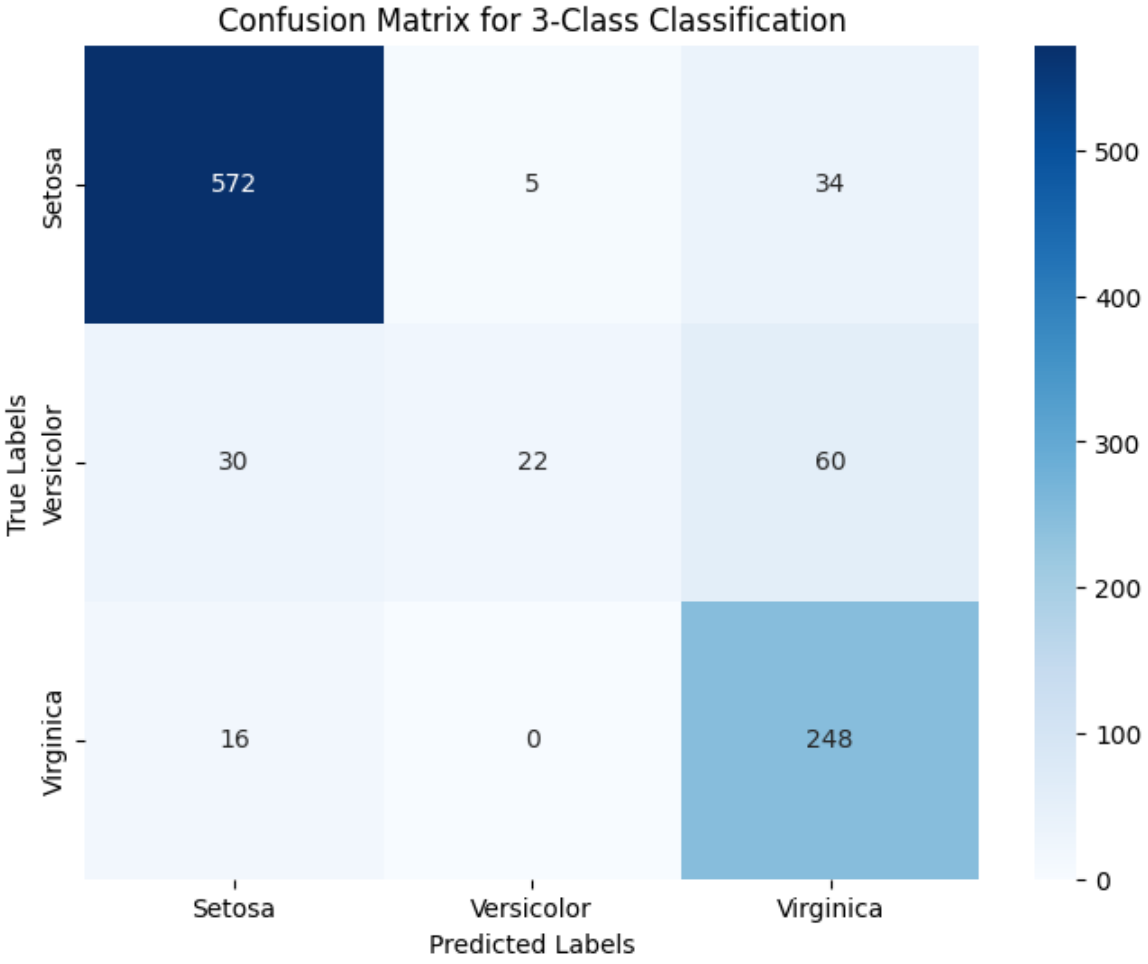
در این مرحله، فیلد هدف را برابر با Type قرار داده و مقادیر x خود را تغییر نمیدهیم. اما لازم به ذکر است که باید فیلد هدف را در ۲ ضرب کنیم تا مقادیر این فیلد اعداد صحیح شوند تا برای کلاس بندی به مشکل نخوریم.

به کمک دستور SVC از کتابخانه sklearn، کار کلاس بندی را انجام میدهیم و خروجی روی داده های تست به صورت زیر خواهد شد:

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.93      0.94      0.93       611
         1.0       0.81      0.20      0.32       112
         2.0       0.73      0.94      0.82       264

    accuracy                           0.85       987
   macro avg       0.82      0.69      0.69       987
weighted avg       0.86      0.85      0.83       987
```



که خروجی قابل قبولی به حساب می آید، بنابراین نیاز به انجام کار اضافه ای برای کلاس بندی الماس ها نداریم و خروجی به دست آمده است.