به نام خدا

1928
K. N. Toosi University of Technology

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

مینی پروژه 1

[سجاد فودازی]

[40007903]

استاد : آقای دکتر مهدی علیاری

لینک مخزن گیت هاب: 5
لینک گوگل کولب پرسش اول: 5
لینک گوگل کولب پرسش دوم: 5
پرسش اول 6
سوال 1.1: 6
سوال 1.2: 8
سوال 1.3: 9
سوال 1.4: 9
سوال 1.5: 10
سوال 1.6: 11
امتیازی 17
پرسش دوم 18
سوال 2.1: 18
سوال 2.2: 19
سوال 2.3: 20
سوال 2.4: 23
سوال 2.5: 24
سوال 2.6: 24
سوال 2.7: 29
امتیازی: 34

تصویر 1 پخش داده برای 5 ستون دلخواه 8
تصویر 2 نقشه حرارتی سوال برای 2 ستون طبقه بندی شده و 3 ستون پیوسته 9
تصویر 3 پخش داده وضعیت خروج از خدمات 10
تصویر 4 - confusion matrix برای داده های تست 14
تصویر 5 - confusion matrix برای داده های ولیدیشن 15
تصویر 6 - confusion matrix برای دیتاهای متعادل شده تست 16
تصویر 8 - confusion matrix برای دیتاهای متعادل شده ولیدیشن 16
تصویر 10 - پخش داده با در نظر گرفتن کلاس های وضعیت خروج از خدمات 17
تصویر 11 - پخش داده دیتای پرسش دوم 18
تصویر 12 - پخش داده های آموزش و آزمون 18
تصویر 13 - مدل آموزش دیده با رگرسیون خطی 21
تصویر 14 - نمودار MSE برای مدل رگرسیون خطی 21
تصویر 15 - نمودار MAE برای مدل رگرسیون خطی 22
تصویر 16 - نمودار R-squared برای مذل رگرسیون خطی 22
تصویر 17 - نمودار خطای MSE به ازای افزایش دیتای آموزش 23
تصویر 18 - نمودار خطای MAE به ازای افزایش دیتای آموزش 23
تصویر 19 - - نمودار خطای R-squared به ازای افزایش دیتای آموزش 24
تصویر 20 - رگرسیون درجه 2 25
تصویر 21 - رگرسیون درجه 3 26
تصویر 22 - رگرسیون درجه 4 26
تصویر 23 - رگرسیون درجه 5 26
تصویر 24 - رگرسیون درجه 6 27
تصویر 25 - نمودار MSE به ازای افزایش توان 27
تصویر 26 - نمودار MAE به ازای افزایش توان 28

تصویر 27 - نمودار R-squared به ازای افزایش توان 28
تصویر 28 - مدل آموزش دیده با الگوریتم درخت تصمیم گیری 30
تصویر 29 – مدل آموزش دیده با الگوریتم جنگل تصادفی 31
تصویر 30 - مدل آموزش دیده با الگوریتم ساپورت وکتور 32
تصویر 31 - MSE سه الگوریتم متفاوت بر روی داده های آموزش و آزمون 32
تصویر 32 – MAE سه الگوریتم متفاوت بر روی داده های آموزش و آزمون 33
تصویر 33 - R-squared سه الگوریتم متفاوت بر روی داده های آموزش و آزمون 33
تصویر 34 - MSE مدل به ازای λ=1 و افزایش توان چندجمله ای 35
تصویر 35 - MAE مدل به ازای λ=1 و افزایش توان چندجمله ای 35
تصویر 36 - R-squared مدل به ازای λ=1 و افزایش توان چندجمله ای 35
تصویر 37 - MSE مدل به ازای λ=1000 و افزایش توان چندجمله ای 36
تصویر 38 - MAE مدل به ازای λ=1000 و افزایش توان چندجمله ای 36
تصویر 39 - R-squared مدل به ازای λ=1000 و افزایش توان چندجمله ای 36

لینک مخزن گیت هاب:

https://github.com/SajjadFdzi8/MachineLearning

لینک گوگل کولب پرسش اول:

https://colab.research.google.com/drive/12kHtTlODCUSPSautUmzXWJlpgz0Kz6iO?usp=sharing

لینک گوگل کولب پرسش دوم:

https://colab.research.google.com/drive/1V8EERy7gCFzySTVCXGKej3mzc4LucKU_?usp=sharing

پرسش اول

سوال 1.1:

مدیر یک بانک به‌شدت نگران افزایش تعداد مشتریانی است که خدمات کارت اعتباری خود را لغو می‌کنند. این موضوع برای بانک بسیار حیاتی شده است، زیرا حفظ مشتریان برای موفقیت کسب‌وکار ضروری است. آنها مشتاق به کارگیری یک راه‌حل پیش‌بینی هستند که بتواند تشخیص دهد کدام مشتریان ممکن است خدمات خود را لغو کنند. با داشتن این اطلاعات، بانک می‌تواند اقدامات پیشگیرانه‌ای انجام دهد تا خدمات بهتری به این مشتریان ارائه دهد و تصمیم آنها را تغییر دهد، در نتیجه نرخ ترک خدمات کاهش یافته و رضایت مشتری افزایش یابد.

این وب‌سایت مجموعه داده‌های متنوعی ارائه می‌دهد و روش‌هایی برای حل مسائل واقعی کسب‌وکار را توضیح می‌دهد. از این سایت به‌عنوان منبعی برای دریافت مجموعه‌داده‌ها و یادگیری استفاده می‌شود. به‌طور کلی، این مجموعه داده شامل حدود 23 ویژگی قابل تحلیل است که اشاره شده است دو ویژگی آخر کاربردی ندارند و بهتر است حذف شوند.

یک نکته قابل توجه در این مجموعه داده، این است که تنها ۱۶.۰۷٪ از مشتریان خدمات خود را لغو کرده‌اند. این موضوع باعث عدم تعادل در مجموعه‌داده می‌شود که کار را برای آموزش مدل پیش‌بینی کمی دشوار می‌کند. این عدم تعادل نیازمند مدیریت دقیق است تا مدل به‌طور مؤثری آموزش ببیند و بتواند مشتریانی که احتمالاً خدمات خود را لغو می‌کنند پیش‌بینی کند، بدون اینکه به دسته اکثریت (مشتریانی که خدمات خود را لغو نکرده‌اند) گرایش پیدا کند.

این پروژه شامل پردازش داده‌ها، مهندسی ویژگی‌ها و آموزش مدل برای پیش‌بینی ترک خدمات مشتریان است و برای بهبود رضایت و وفاداری مشتریان از اهمیت ویژه‌ای برخوردار است.

این دیتا شامل 23 ستون است که دو ستون آخر طبق گفته وبسایت استفاده نمیشوند:

1- شماره مشتری(یکتا)
2- وضعیت خروج از خدمات(بسته: 1 و باز: 0)
3- سن مشتری
4- جنسیت(مرد: M و زن: F)
5- تعداد افراد تحت تکفل
6- سطح تحصیلات
7- وضعیت تأهل

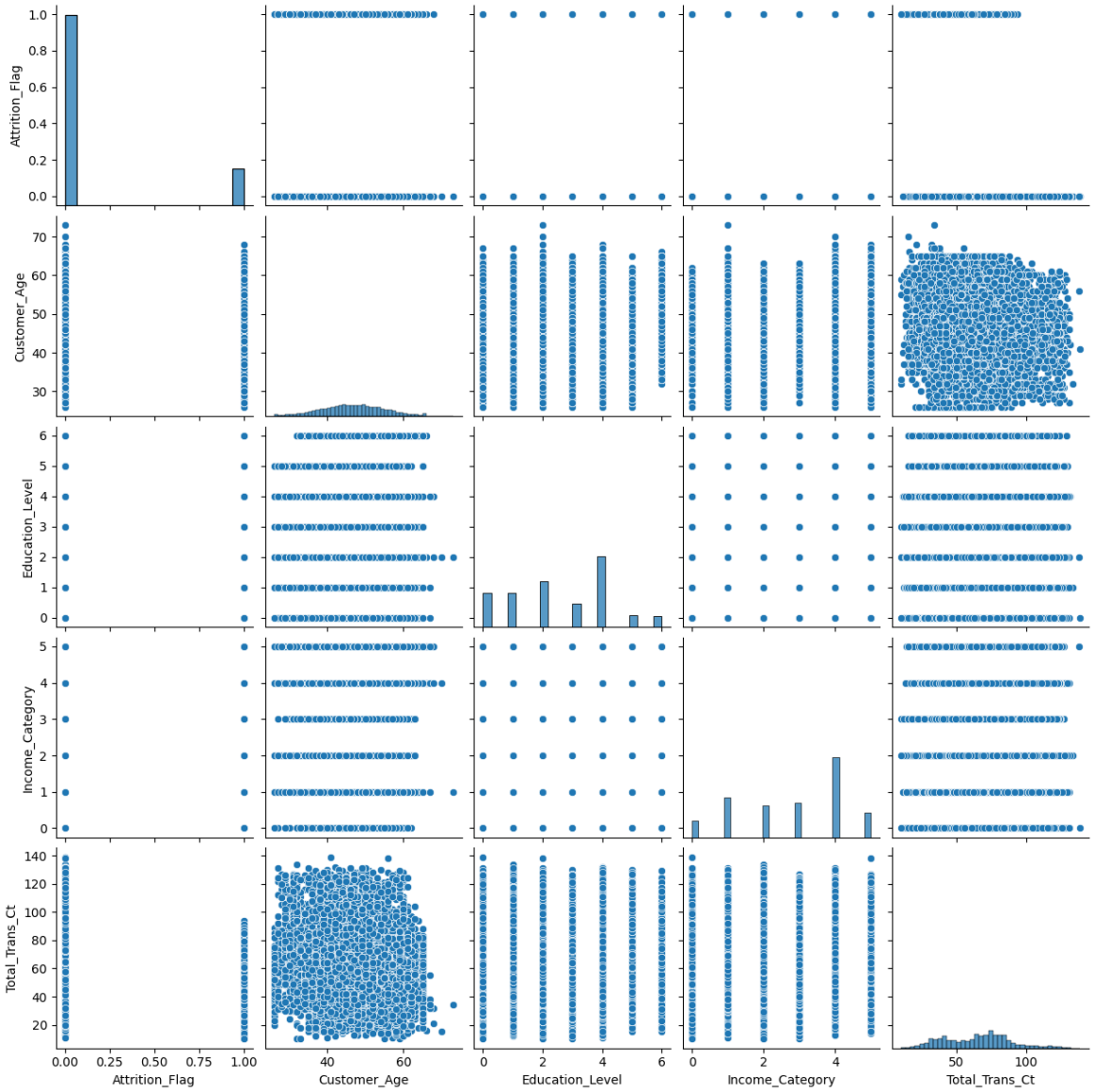
8- دسته‌بندی درآمد
9- دسته‌بندی کارت
10- تعداد ماه‌ها از عضویت
11- تعداد کل تعاملات
12- تعداد ماه‌های غیرفعال در 12 ماه گذشته
13- تعداد تماس‌ها در 12 ماه گذشته
14- سقف اعتبار
15- کل مانده اعتباری چرخشی
16- میانگین اعتبار قابل استفاده
17- تغییر کل مبلغ از فصل چهارم به فصل اول
18- کل مبلغ تراکنش‌ها
19- تعداد کل تراکنش‌ها
20- تغییر تعداد کل تراکنش‌ها از فصل چهارم به فصل اول
21- نسبت میانگین استفاده
22- خروجی اول از مدل طبقه‌بندی بیز ساده
23- خروجی دوم از مدل طبقه‌بندی بیز ساده

10100 نمونه در این وبسایت موجود است.

سوال 1.2:

برای پلات پخش داده از داده های موجود در ستون های زیر استفاده میکنیم:

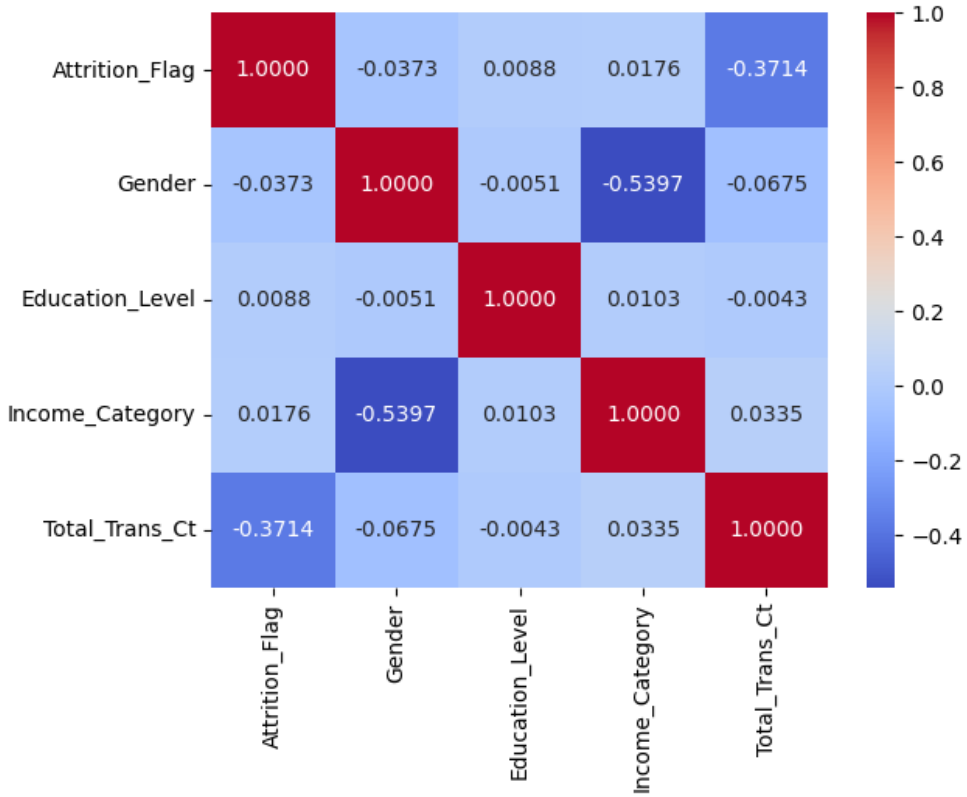
وضعیت خروج از خدمات، سن مشتری، سطح تحصیلات، دسته‌بندی درآمد، تعداد کل تراکنش‌ها



تصویر 1 پخش داده برای 5 ستون دلخواه

سوال 1.3:

با استفاده از ستون های طبقه بندی شده(وضعیت خروج از خدمات، جنسیت) و ستون های پیوسته(سطح تحصیلات، دسته‌بندی درآمد، تعداد کل تراکنش‌ها) همبستگی میان آن ها را با نقشه حرارتی پلات میکنیم:



تصویر 2 نقشه حرارتی سوال برای 2 ستون طبقه بندی شده و 3 ستون پیوسته

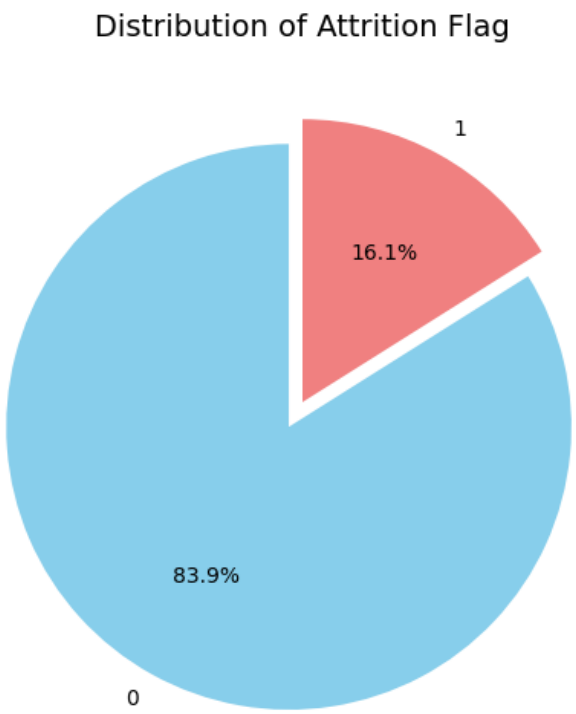
با توجه به نقشه مشاهده میشود که همبستگی خوبی میان دسته بندی درآمد و جنسیت مشتری ها وجود دارد. همچنین بدیهی است که تعداد تراکنش ها به باز یا بسته بودن حساب مشتری ها وابسته باشد. بقیه ویژگی ها با یکدیگر همبستگی خاصی ندارند.

سوال 1.4:

با استفاده از data.isnull().values.any() میتوانیم ببینیم که دیتای NAN وجود ندارد.

سوال 1.5:

این ویژگی دارای دو کلاس است که همانطور که در قسمت اول اشاره شد به کسانی که حساب خود را بسته اند لیبل 1 زده شده و به کسانی که حساب آن ها هنوز باز است لیبل 0 زده شده است.



تصویر 3 پخش داده وضعیت خروج از خدمات

عدم تعادل در داده‌ها می‌تواند تأثیر قابل توجهی روی عملکرد مدل نهایی داشته باشد. این مسئله به ویژه زمانی که داده‌ها به طور نامتقارن بین کلاس‌های مختلف توزیع شده باشند (مثلاً تعداد داده‌های یک کلاس بسیار بیشتر از دیگری باشد)، مدل به سمت کلاس غالب گرایش پیدا میکند و دقت پایین‌تری برای کلاس‌های دیگر داشته باشد. این امر می‌تواند پیش‌بینی‌های نادرستی ایجاد کند.

برای رفع مشکل عدم تعادل، راهکارهای زیر می‌تواند موثر باشد:

1. نمونه‌برداری مجدد
2. Oversampling (تکنیک SMOTE)
3. Undersampling
4. Weighted Loss Functions
5. استفاده از معیارهای ارزیابی مناسب: معیارهایی مانند F1Score، Precision، و Recall به جای صرفاً Accuracy.

باید مجموعه داده‌ها را به سه بخش تقسیم کنید:

1. داده‌های آموزشی (Training Data): 80 درصد دیتا
2. داده‌های آزمون (Testing Data): 15 درصد دیتا(1519)
3. داده‌های اعتبارسنجی (Validation Data): 5 درصد دیتا(507)

اگر داده‌ها به درستی تقسیم نشوند، ممکن است عملکرد مدل روی داده‌هایی که قبلاً دیده است، بیش از حد خوش‌بینانه به نظر برسد و مدل در دنیای واقعی عملکرد ضعیفی داشته باشد. بنابراین، تقسیم‌بندی مناسب باعث می‌شود که مدل به طور مستقل روی داده‌های آزمون ارزیابی شود و عملکرد واقعی آن مشخص گردد.

سوال 1.6:

با استفاده از Y = data.iloc[:, 1] ستون وضعیت خروج از خدمات را بعنوان خروجی تعریف میکنیم و با استفاده از X = data.drop(columns=data.columns[1]) و دراپ کردن ستون اول بقیه ستون ها را بعنوان ورودی تعریف میکنیم. باید داده های تست و ولیدیشن را نیز در همین ابتدا جدا کنیم که با استفاده از کد زیر ابتدا 20 درصد داده را تست میگیریم و سپس 25 درصد داده های تست را ولیدیشن میگیریم:

```
indices = Y.index
train_indices, test_indices = train_test_split(indices, test_size=0.2, stratify=Y, random_state=93)
X_train, Y_train = X.loc[train_indices], Y.loc[train_indices]
X_test, Y_test = X.loc[test_indices], Y.loc[test_indices]
test_indices, val_indices = train_test_split(test_indices, test_size=0.25, random_state=93)
X_test, Y_test = X.loc[test_indices], Y.loc[test_indices]
X_val, Y_val = X.loc[val_indices], Y.loc[val_indices]
```

رندوم استیت باید دو عدد آخر شماره دانشجویی باشد که چون 03 میشود بجای 0 از 9 که عدد قبلی آن است استفاده شده است. تعداد داده های تست 1519 و تعداد داده های ولیدیشن 507 تا شده است.

حال نوبت اموزش دادن مدل با داده های train است. برای اینکار با کمک گرفتن از chatgpt و استفاده از LogisticRegression، StandardScaler، GridSearchCV و accuracy_score که در کتابخانه sklearn هستند کدی را ران میکنیم که برای ما بهترین penalty ها را مشخص کند. با این کد بین پنالتی های موجود که ممکن است در این مسئله غیر بالانس به کار آیند را تشخیص میدهیم و سپس مدل را بر اساس آن ها آموزش میدهیم. کد استفاده شده:

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
param_grid = {
    'penalty': ['l1', 'l2', 'elasticnet'],
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'saga'],
    'tol': [1e3, 1e4, 1e5],
    'class_weight': ['balanced', None],
}
grid_search = GridSearchCV(LogisticRegression(max_iter=2000, random_state=93),
param_grid, cv=5, n_jobs=1)
grid_search.fit(X_train_scaled, Y_train)
best_model = grid_search.best_estimator_
Y_pred = best_model.predict(X_test_scaled)
Y_val_pred = best_model.predict(X_val_scaled)
print(f"Best Parameters: {grid_search.best_params_}")
```

penalty:

- l1: جریمه بر اساس قدر مطلق ضرایب (Lasso).
- l2: جریمه بر اساس مربع ضرایب (Ridge).
- elasticnet: ترکیبی از l1 و l2.

C: برعکس قدرت جریمه: مقدار کوچک‌تر C جریمه را قوی‌تر می‌کند (مدل ساده‌تر و مقاوم‌تر به بیش‌برازش)، و مقدار بزرگ‌تر C جریمه را کاهش می‌دهد (مدل پیچیده‌تر).

solver:

- liblinear: مخصوص مدل‌های کوچک و استفاده با l1 و l2.

- saga: برای مجموعه داده‌های بزرگ و پشتیبانی از l2 ,l1, و elasticnet.

tol: مقدار عددی که دقت مورد نیاز برای توقف بهینه‌سازی را تعیین می‌کند. مقدار کوچک‌تر دقت بالاتری را تضمین می‌کند اما زمان محاسبات را افزایش می‌دهد.

class_weight:

- balanced: به طور خودکار وزن کلاس‌ها را متناسب با توزیع داده تنظیم می‌کند.
- None: وزن برابر برای همه کلاس‌ها در نظر گرفته می‌شود.

GridSearchCV: برای پیدا کردن بهترین ترکیب پارامترهای مدل از طریق آزمون و خطا استفاده می‌شود. همه ترکیب‌های ممکن پارامترهای تعریف‌شده در param_grid را آزمایش می‌کند.

LogisticRegression(max_iter=2000, random_state=93):

- max_iter=2000: حداکثر تعداد تکرارها برای همگرایی الگوریتم را تنظیم می‌کند (2000 بار).
- random_state=93: مقدار ثابتی برای تولید نتایج قابل تکرار تنظیم می‌کند.
- param_grid: مجموعه‌ای از پارامترهای قابل تنظیم است (مثل penalty, C, و غیره) که برای یافتن بهترین ترکیب بررسی می‌شوند.

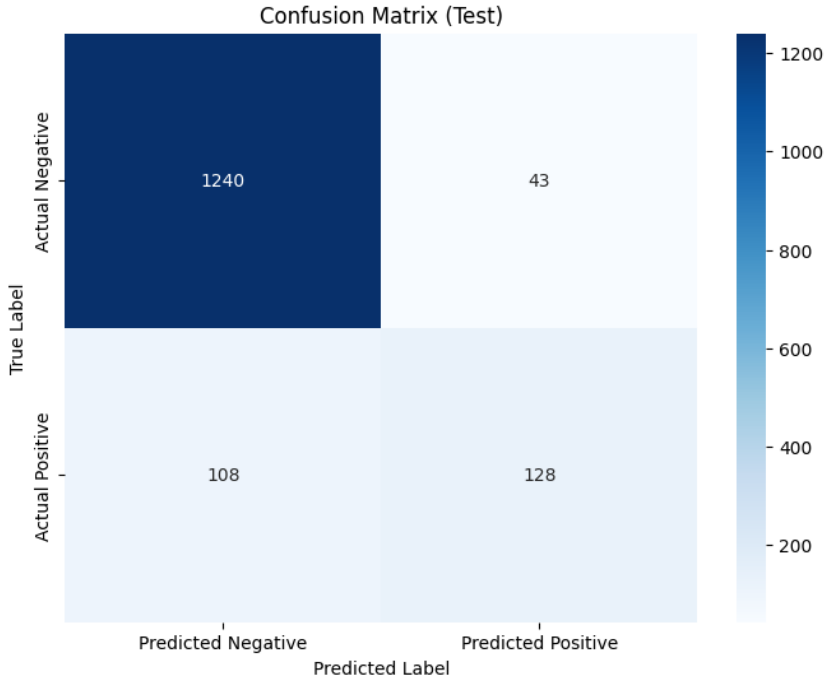
cv=5: تعداد بخش‌ها (Fold) در اعتبارسنجی متقابل (Cross-Validation).

n_jobs=-1: از تمام هسته‌های پردازنده برای انجام محاسبات به صورت موازی استفاده می‌کند تا زمان اجرا کاهش یابد.

در نهایت مقادیر زیر برای آموزش استفاده میشوند:

Best Parameters: {'C': 0.1, 'class_weight': None, 'penalty': 'l2', 'solver': 'liblinear', 'tol': 0.001}

حالا برای بررسی نوع عملکرد برای داده های تست و ولیدیشن که جزوی از مرحله آموزش نبوده اند میتوانیم راهکار های متفاوتی استفاده کنیم. قبل از هرچیزی بهتر است confusion matrix را برای تست و ولیدیشن ببینیم.



تصویر 4 - confusion matrix برای داده های تست

حال با استفاده از مقادیر این ماتریس که میتوان TP، TN، FP و FN را مشاهده کرد میتوانیم f1، precission، recall و balance را محاسبه کنیم:

F1 Score test: 0.89

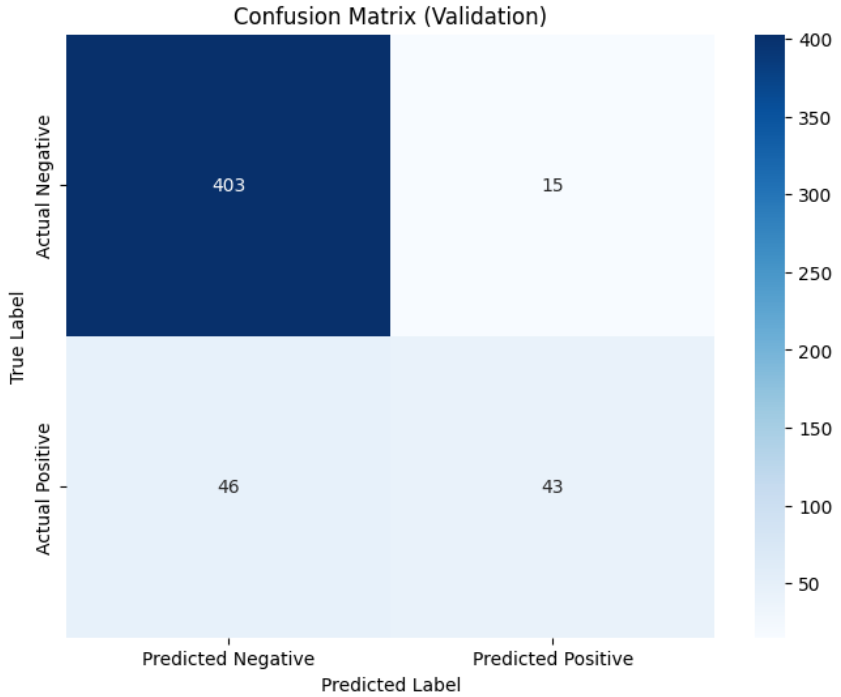
Precision Test: 0.748

Recall Test: 0.542

Balanced Accuracy Test: 0.754

مشاهده میشود که 108 داده را اشتباه منفی تشخیص داده ایم که خوب نیست ولی نسبت به کل داده های تست مقدار خوبی برای ما دارد. بصورت کلی مدل به خوبی آموزش داده شده و بر روی داده های تست جواب میدهد.

برای مقادیر ولیدیشن هم میتوانیم این کار ها انجام دهیم.



تصویر 5 - confusion matrix برای داده های ولیدیشن

F1 Score Validation: 0.87

Precision Validation: 0.741

Recall Validation: 0.483

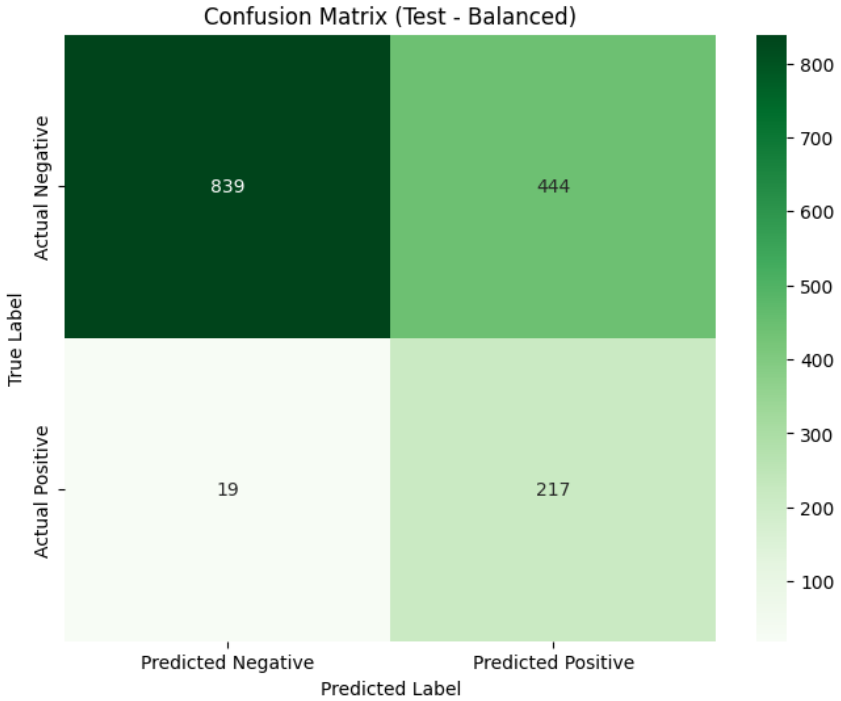
Balanced Accuracy Validation: 0.723

تفاوت آنچنانی با مقادیر تست وجود ندارد که این نتیجه خوبی است.

برای بخش دوم که خواسته شده ابتدا دیتا را متعادل سازی کنیم از راهکار smote استفاده میکنیم. این راهکار داده های مصنوعی تولید میکند و ما برای داده هایی که تعداد آن ها کمتر است از این راهکار استفاده میکنیم سپس همان مراحل قبل را طی میکنیم. بهترین پارامتر ها طبق کد تولید شده:

Best Parameters (Balanced Data): {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}

نتایج به شکل زیر خواهد بود:

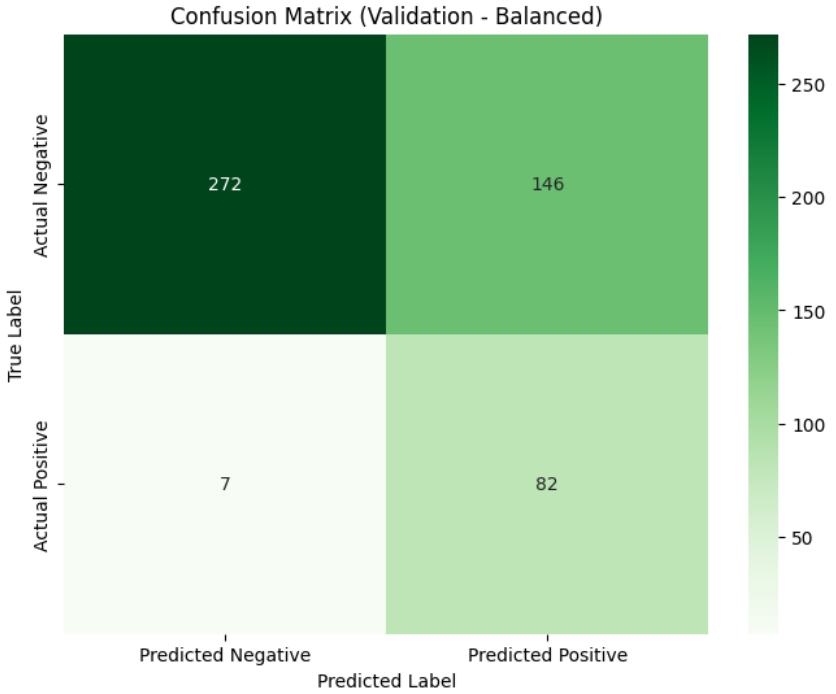


تصویر 6 - confusion matrix برای دیتاهای متعادل شده تست

F1 Score test: 0.89

Precision Test: 0.89

Recall Test: 0.9



تصویر 7 - confusion matrix برای دیتاهای متعادل شده ولیدیشن

F1 Score Validation: 0.74

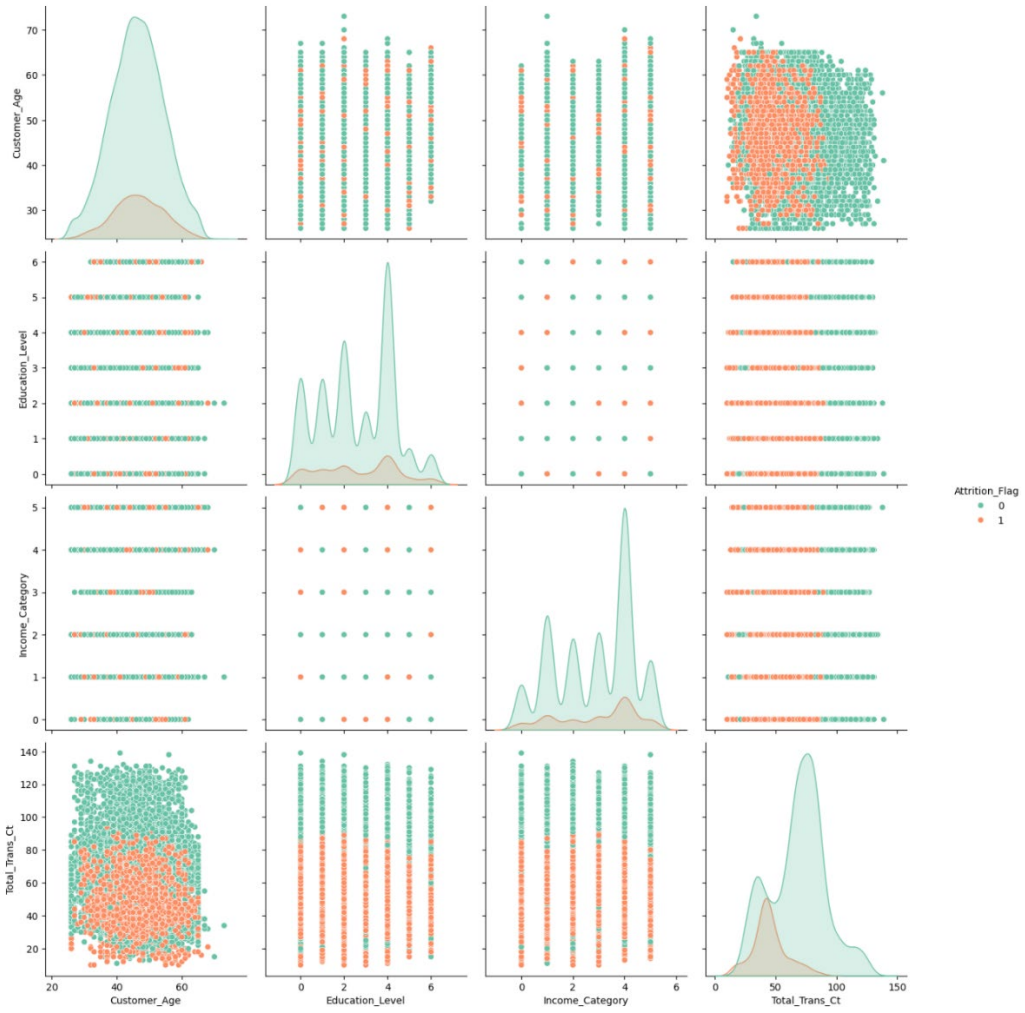
Precision Validation: 0.88

Recall Validation: 0.70

به وضوح رویت میشود که وقتی دیتا متعادل نشده بود نوع آموزش بهتر جواب داده است و استفاده از SMOTE آنطور که انتظار داشتیم جواب نداده است. البته باید دیتای متعادل شده نتیجه بهتری میداد که یعنی راهکار ما کاملا درست نبوده است. همچنین نکته ای باید به آن دقت شود این است که مدل آموزش دیده در تشخیص دیتایی که از اول کمتر بود و قصد داشتیم با روش SMOTE تعدادش رو زیاد بکنیم اصلا خوب عمل نمیکند.

امتیازی

همان 5 ستونی که در قسمت قبل گرفته بودیم در نظر میگیریم ولی این دفعه تفاوت بین 0 و 1 بودن کلاس وضعیت خروج از خدمات قائل میشویم و نتیجه به شکل زیر میشود:

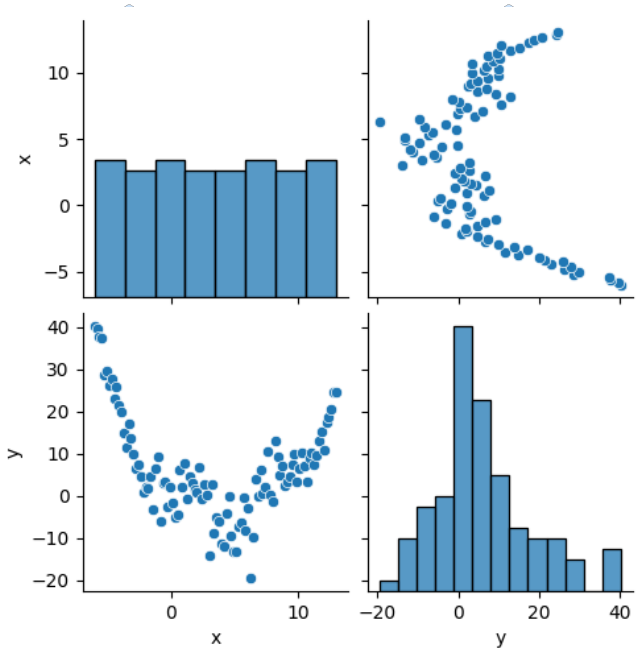


تصویر 8 - پخش داده با در نظر گرفتن کلاس های وضعیت خروج از خدمات

پرسش دوم

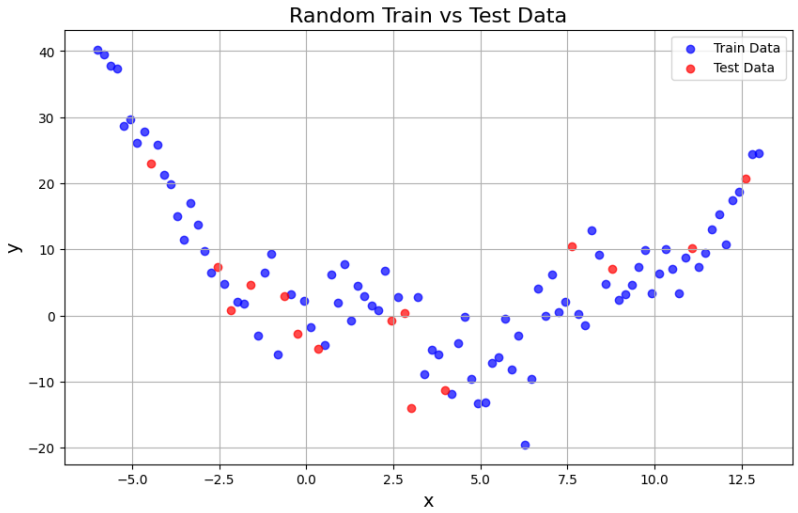
سوال 2.1:

ابتدا داده های یک بعدی داده شده را به محیط کدنویسی کولب منتقل کردیم سپس با استفاده از np.linspace(-6, 13, len(y_data)) داده ها را دو بعدی کردیم. پخش داده بدست آمده:



تصویر 9 - پخش داده دیتای پرسش دوم

85 درصد داده ها را بعنوان داده آموزش و 15 درصد آن ها را بعنوان آزمون انتخاب میکنیم. برای انتخاب داده های آزمون 15 درصد را بصورت رندوم انتخاب میکنیم.



تصویر 10 - پخش داده های آموزش و آزمون

سوال 2.2:

1. میانگین مربعات خطا (Mean Squared Error - MSE): میانگین مربع اختلاف بین مقادیر پیش‌بینی شده توسط مدل و مقادیر واقعی است. MSE هرچه کمتر باشد، نشان‌دهنده عملکرد بهتر مدل است. از آنجایی که خطاها به توان دو می‌رسند، MSE به خطاهای بزرگ حساس‌تر است.

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

- n: تعداد نمونه‌ها
- y_i: مقدار واقعی متغیر وابسته برای نمونه i
- $\hat{y}_i$: مقدار پیش‌بینی شده توسط مدل برای نمونه i

2. میانگین قدر مطلق خطا (Mean Absolute Error - MAE): میانگین قدر مطلق اختلاف بین مقادیر پیش‌بینی شده و مقادیر واقعی است. MAE نسبت به MSE به خطاهای بزرگ حساسیت کمتری دارد. بنابراین، اگر وجود چند خطای بزرگ در مدل برای شما مهم نباشد، MAE می‌تواند معیار مناسبی باشد.

$$MAE = \frac{1}{n}\sum|y_i - \hat{y}_i|$$

3. R-squared (R^2): نشان می‌دهد که چه مقدار از تغییرات متغیر وابسته توسط مدل توضیح داده شده است. R-squared بین 0 تا 1 متغیر است. هرچه مقدار R-squared به 1 نزدیک‌تر باشد، نشان‌دهنده برازش بهتر مدل بر روی داده‌ها است. R-squared همیشه با افزایش تعداد متغیرهای مستقل افزایش می‌یابد. بنابراین، برای مقایسه مدل‌های با تعداد متغیرهای مختلف، بهتر است از معیارهای دیگری مانند Adjusted R-squared استفاده شود.

بصورت کلی:

- MSE برای زمانی مناسب است که خطاهای بزرگ بسیار مهم باشند.
- MAE برای زمانی مناسب است که وجود چند خطای بزرگ اهمیت چندانی نداشته باشد.
- R-squared برای ارزیابی کلی برازش مدل مناسب است.

سوال 2.3:

قبل از آموزش دیتا ها را نرمالایز میکنیم. نرمال‌سازی داده‌ها یک پیش‌پردازش ضروری در بسیاری از مدل‌های یادگیری ماشین است، به خصوص در موارد زیر:

1. عدم تناسب مقیاس داده‌ها: اگر مقادیر x و y در بازه‌های مختلفی قرار داشته باشند. مقدار بزرگتر می‌تواند تأثیر نامتناسبی روی محاسبات داشته باشد. نرمال‌سازی این مشکل را حل می‌کند و مقادیر همه ویژگی‌ها را در یک مقیاس استاندارد (میانگین 0 و انحراف معیار 1) قرار می‌دهد.
2. پایداری عددی: در محاسبات ماتریسی، وجود مقادیر خیلی بزرگ یا خیلی کوچک می‌تواند باعث ناپایداری عددی شود و دقت محاسبات را کاهش دهد. نرمال‌سازی این پایداری را افزایش می‌دهد.
3. افزایش کارایی مدل: وقتی داده‌ها نرمال می‌شوند، مدل سریع‌تر همگرا می‌شود و دقت پیش‌بینی افزایش می‌یابد.

در این مسئله، داده‌های x و y ممکن است مقیاس‌های متفاوتی داشته باشند، و از آنجایی که رگرسیون خطی مستقیماً به این مقیاس‌ها حساس است، نرمال‌سازی باعث شده است که مدل به درستی شیب و عرض از مبدأ را تخمین بزند. پس از محاسبه مقادیر بهینه در مقیاس نرمال‌شده، مقادیر پیش‌بینی‌شده را به مقیاس اصلی بازگرداندیم تا با داده‌های واقعی قابل مقایسه باشند.

رگرسیون خطی یکی از ساده‌ترین و در عین حال قدرتمندترین مدل‌های پیش‌بینی است که بر اساس ایده تطبیق یک خط مستقیم به داده‌ها عمل می‌کند. هدف این است که رابطه بین متغیر مستقل (x) و متغیر وابسته (y) را به شکلی مدل کنیم که اختلاف بین مقادیر پیش‌بینی‌شده و مقادیر واقعی حداقل شود.

فرض می‌کنیم که رابطه بین x و y به صورت خطی است:

$$y = wx + b$$

هدف ما یافتن w و b به گونه‌ای است که خطای کل به حداقل برسد. برای این کار از فرمول‌های بسته (Closed-form) استفاده کردیم:

$$w = \frac{\sum(x_i - x^-)(y_i - y^-)}{\sum(x_i - x^-)^2}$$

$$b = y^- - wx^-$$

این فرمول‌ها به ما اجازه می‌دهند که بدون نیاز به الگوریتم‌های پیچیده یا کتابخانه‌های آماده، مقادیر بهینه w و b را به دست آوریم.

با استفاده از همین مدل رگرسیون و نرمالایز کردن مقادیر زیر بدست می آیند.

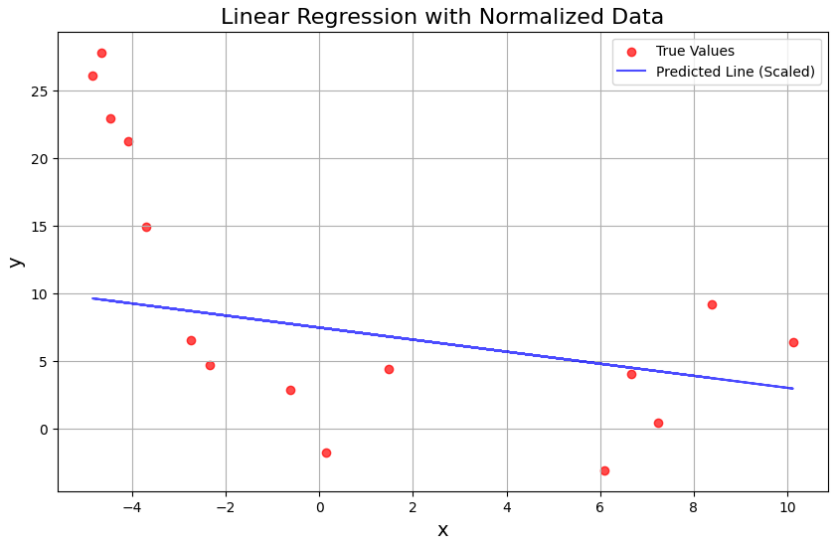
Scaled Weight (w): -0.1984

Scaled Bias (b): 5.2325e-17

Mean Squared Error (MSE): 81.1936

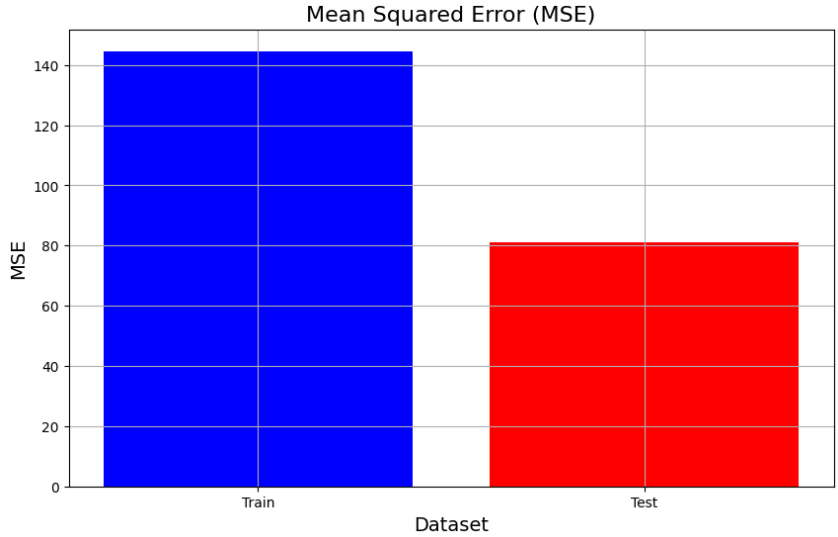
Mean Absolute Error (MAE): 7.3041

R-squared: 0.17147

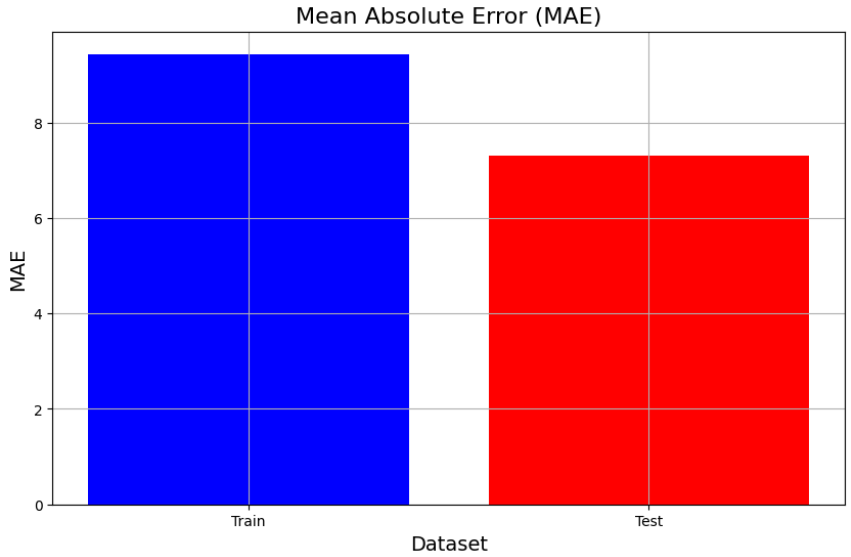


تصویر 11 - مدل آموزش دیده با رگرسیون خطی

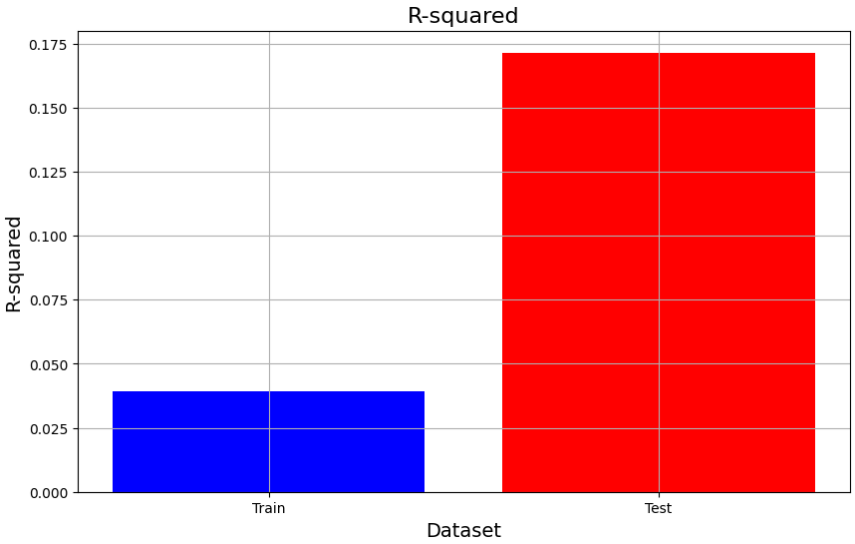
بدیهی است که استفاده از رگرسیون خطی برای آموزش این دیتاها که شکل خطی ندارند نتیجه مطلوبی ندارد مگر اینکه تعداد ویژگی ها بیشتر شود که در اینجا فقط x و y داریم.



تصویر 12 - نمودار MSE برای مدل رگرسیون خطی



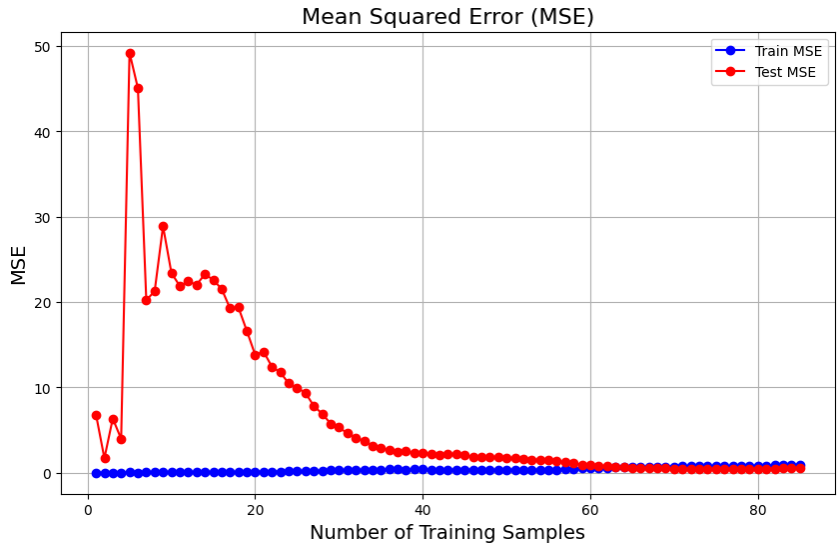
تصویر 13 - نمودار MAE برای مدل رگرسیون خطی



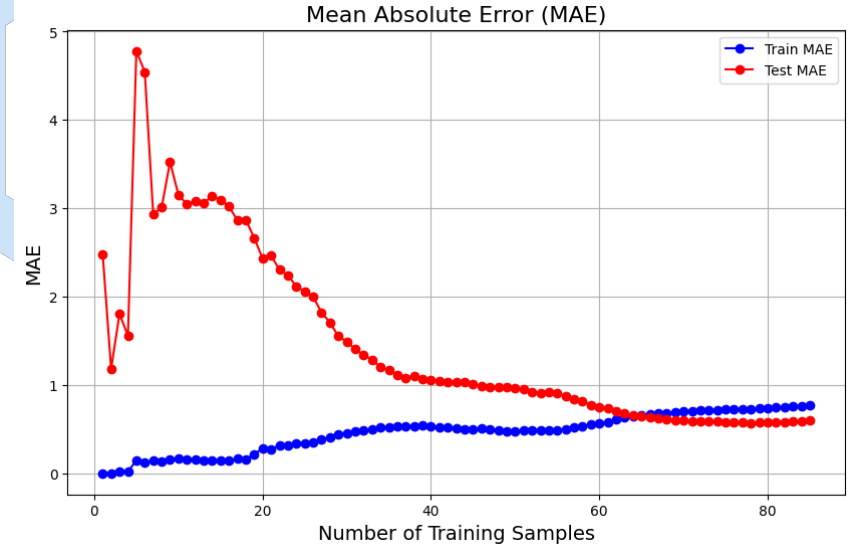
تصویر 14 - نمودار R-squared برای مدل رگرسیون خطی

سوال 2.4:

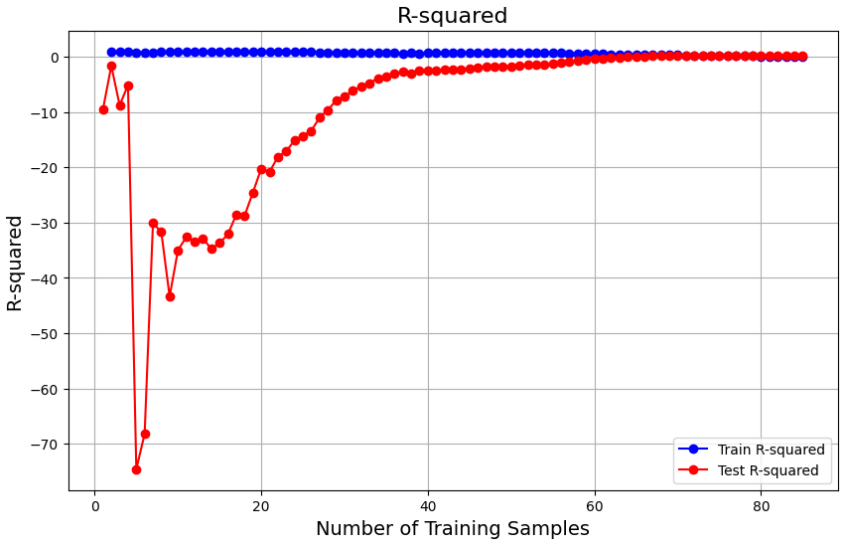
برای این بخش نیاز به سه نمودار برای بررسی تاثیر افزایش نمونه برای بهتر شدن مدل آموزش دیده شده داریم.



تصویر 15 - نمودار خطای MSE به ازای افزایش دیتای آموزش



تصویر 16 - نمودار خطای MAE به ازای افزایش دیتای آموزش



تصویر 17 - - نمودار خطای R-squared به ازای افزایش دیتای آموزش

Train Error (آبی): همان‌طور که مشاهده می‌شود، خطای داده‌های آموزشی بسیار پایین است و با افزایش داده‌های آموزشی ثابت می‌ماند.

Test Error (قرمز): خطای داده‌های آزمون در ابتدا بالا است و با افزایش داده‌های آموزشی کاهش پیدا می‌کند. این نشان می‌دهد که مدل با داده‌های بیشتر بهتر تعمیم پیدا می‌کند.

سوال 2.5:

اگر خطای مدلی 10 باشد یعنی این مدل underfit است و با افزایش داده ها میتواند عملکرد خیلی بهتری ارائه بدهد. البته باید توجه داشت که زیاد شدن دیتا ممکن است باعث شود مدل overfit شود که در صورت بروز این اتفاق حتی عملکرد بدتر هم میشود. بصورت خلاصه میتوانیم عملکرد مدل را با افزایش دیتاها بهتر کنیم اما قطعا نمیتوانیم از خطای انسان بهتر باشیم چون انسان توانایی درک پیچیدگی ها و مسائل از قبل تعریف نشده را دارد. برتری مدل میتواند این باشد که کم هزینه تر از انسان و سریعتر از انسان است و میتواند با یک خطای محدودی نتیجه مطلوب ما را بدهد.

سوال 2.6:

ایجاد ویژگی‌های چندجمله‌ای (Adding Polynomial Features): در هر تکرار (برای هر درجه از چندجمله‌ای)، ستون‌هایی ایجاد می‌شود که توان‌های مختلف متغیر x را شامل می‌شوند. به عنوان مثال:

درجه 1: x

درجه 2: x و x^2

به همین ترتیب تا درجه 6 جلو میریم. این ویژگی‌ها به صورت دستی با استفاده از np.column_stack() ساخته شده‌اند.

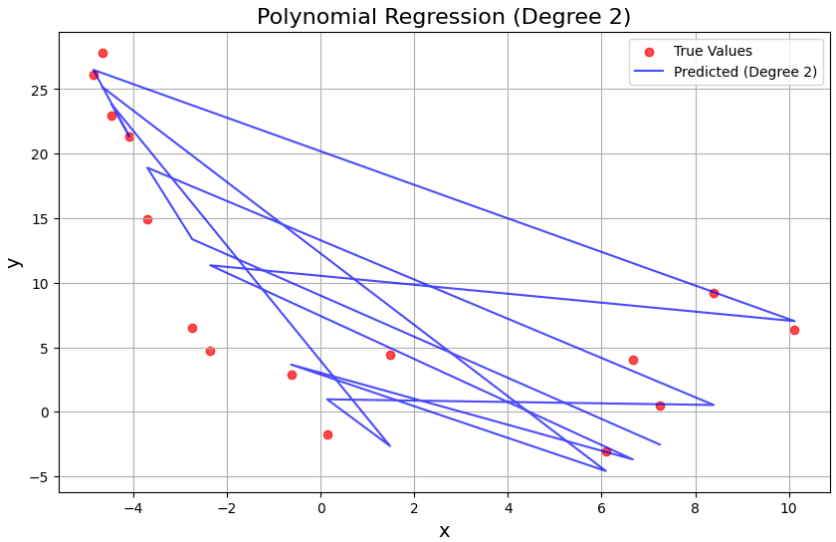
برای محاسبه ضرایب مدل به صورت دستی و با استفاده از روش کمترین مربعات خطا (Least Squares) آموزش داده می‌شود:

$$w = (XX^T)^{-1}Y^TX$$

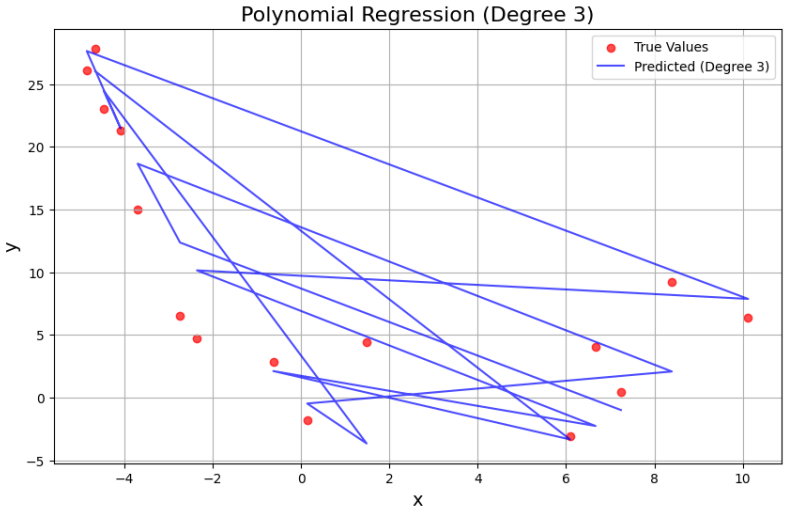
همچنین، مقدار بایاس صورت جداگانه محاسبه می‌شود:

$$b = Y_{mean} - \sum wX_{mean}$$

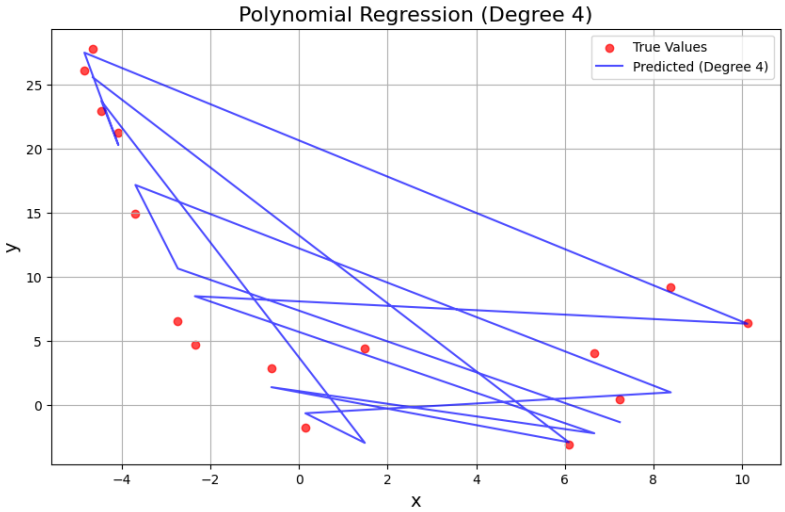
هر درجه جدید از چندجمله‌ای، انعطاف‌پذیری مدل را افزایش می‌دهد. این امر باعث می‌شود مدل بتواند به الگوهای پیچیده‌تر در داده‌ها بپردازد. اما اگر درجه بیش از حد بالا باشد، احتمال Overfitting افزایش می‌یابد، زیرا مدل بیش از حد به داده‌های آموزش نزدیک می‌شود و عملکرد روی داده‌های تست کاهش می‌یابد.



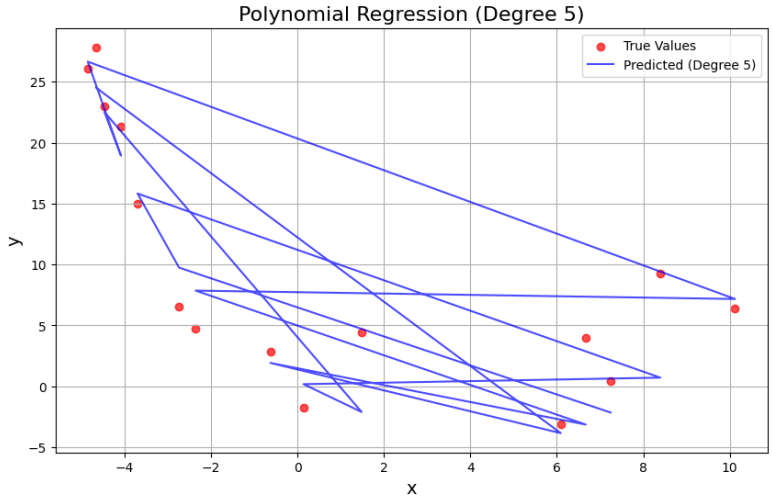
تصویر 18 - رگرسیون درجه 2



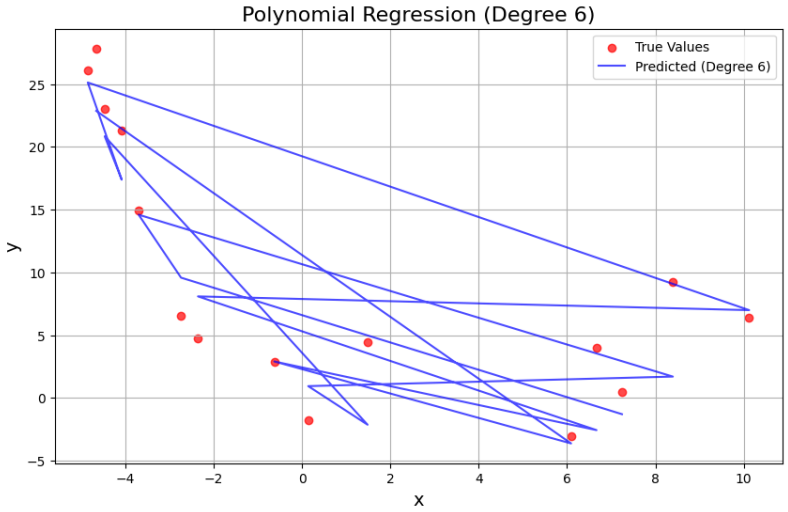
تصویر 19 - رگرسیون درجه 3



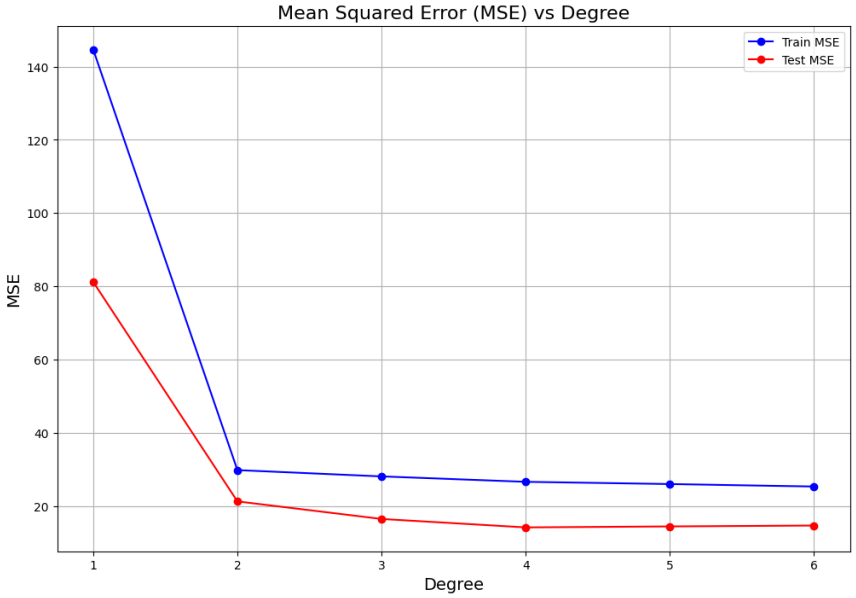
تصویر 20 - رگرسیون درجه 4



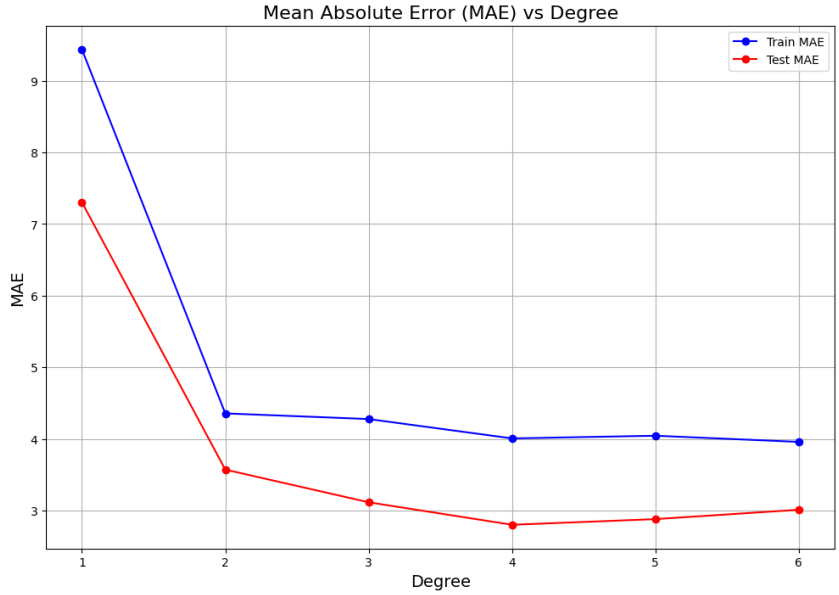
تصویر 21 - رگرسیون درجه 5



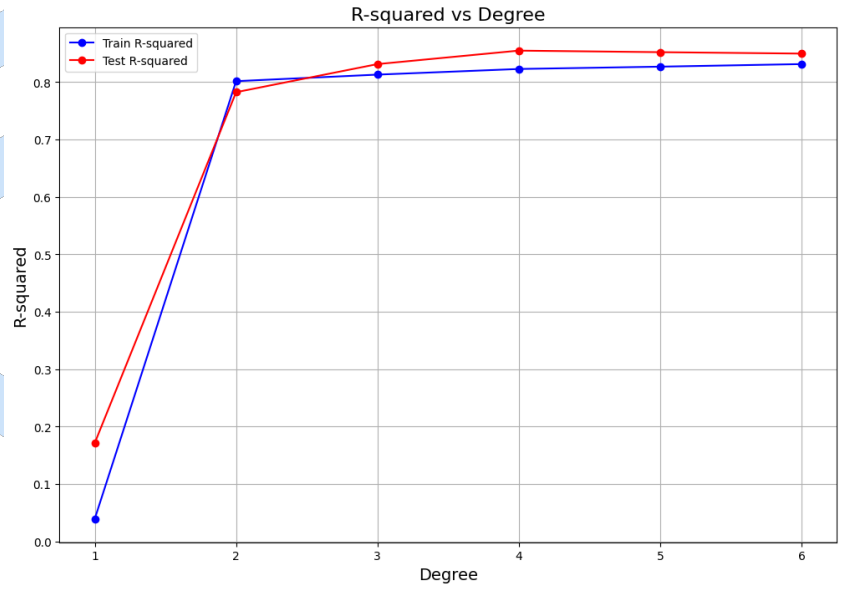
تصویر 22 - رگرسیون درجه 6



تصویر 23 - نمودار MSE به ازای افزایش توان



تصویر 24 - نمودار MAE به ازای افزایش توان



تصویر 25 - نمودار R-squared به ازای افزایش توان

همانطور که در نمودار ها قابل مشاهده است از توان 4 به بعد دیگر خطا تغییری نمیکند و ثابت میماند و این نشان میدهد که اگر توان را بیشتر کنیم مدل overfit میشود و نتیجه بدتری ارائه میدهد. تا توان چهارم میتواند مدل را به خوبی آموزش دهد و نتیجه بسیاری خوبی دریافت کرد.

سوال 2.7:

1. Decision Tree (درخت تصمیم‌گیری): درخت تصمیم‌گیری الگوریتمی است که داده‌ها را به صورت بازگشتی به زیرمجموعه‌هایی تقسیم می‌کند تا یک هدف (مانند پیش‌بینی یک مقدار) را به بهترین شکل ممکن پیش‌بینی کند. هر گره در درخت یک شرط یا ویژگی را بررسی می‌کند و بر اساس مقدار آن ویژگی، داده‌ها را به شاخه‌های مختلف تقسیم می‌کند.

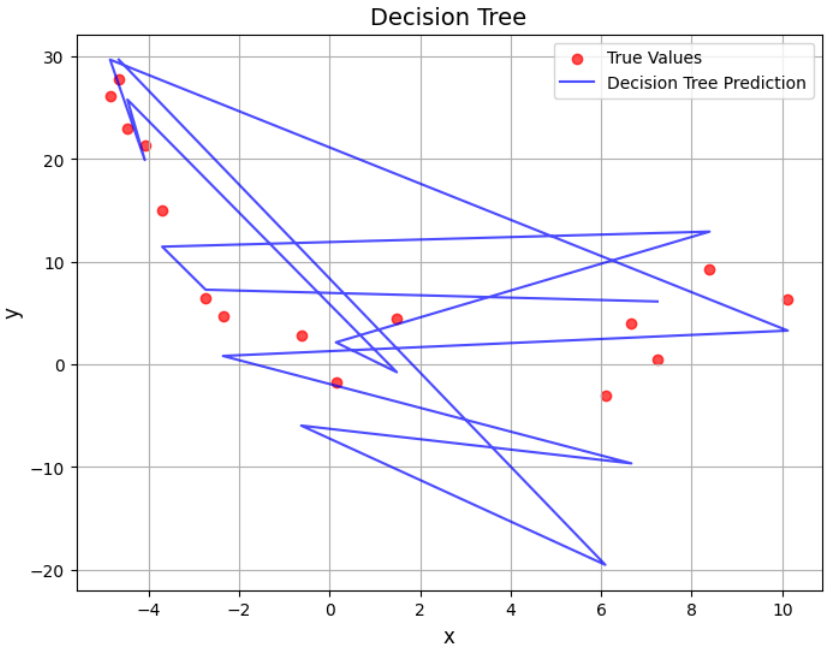
ویژگی‌ای را که بهترین تقسیم را فراهم می‌کند (با کاهش بیشترین خطا، مثلاً بر اساس معیار MSE در رگرسیون) انتخاب می‌کند. داده‌ها را بر اساس شرط انتخابی تقسیم می‌کند. این فرآیند را بازگشتی ادامه می‌دهد تا به یک شرط توقف برسد (مثلاً تعداد نمونه‌های یک گره کم‌تر از مقدار مشخصی شود یا عمق درخت از مقدار معینی تجاوز کند).

مزایا:

- تفسیرپذیری بالا (می‌توان مسیر تصمیم را به سادگی دنبال کرد).
- مناسب برای داده‌های غیرخطی و نامتقارن.
- نیازی به نرمال‌سازی یا استانداردسازی داده ندارد.

معایب:

- ممکن است Overfitting رخ دهد (درخت بسیار پیچیده می‌شود و به داده‌های آموزش بیش از حد نزدیک می‌شود).
- حساسیت زیاد به تغییرات کوچک در داده (درخت‌های مختلف ممکن است برای داده‌های اندکی متفاوت ایجاد شوند).



تصویر 26 - مدل آموزش دیده با الگوریتم درخت تصمیم گیری

2. Random Forest (جنگل تصادفی): الگوریتمی مبتنی بر بگینگ (Bagging) است که از ترکیب چندین درخت تصمیم‌گیری (غیرمرتبط) برای بهبود دقت استفاده می‌کند. هر درخت روی یک زیرمجموعه تصادفی از داده‌ها (با جایگذاری) آموزش می‌بینند، و در رگرسیون، خروجی نهایی برابر با میانگین خروجی‌های تمام درخت‌ها است. برای ساخت هر درخت، الگوریتم ویژگی‌های کاندیدای تقسیم را به صورت تصادفی زیرنمونه‌برداری می‌کند (Random Feature Selection). این ویژگی باعث کاهش همبستگی بین درخت‌ها می‌شود.

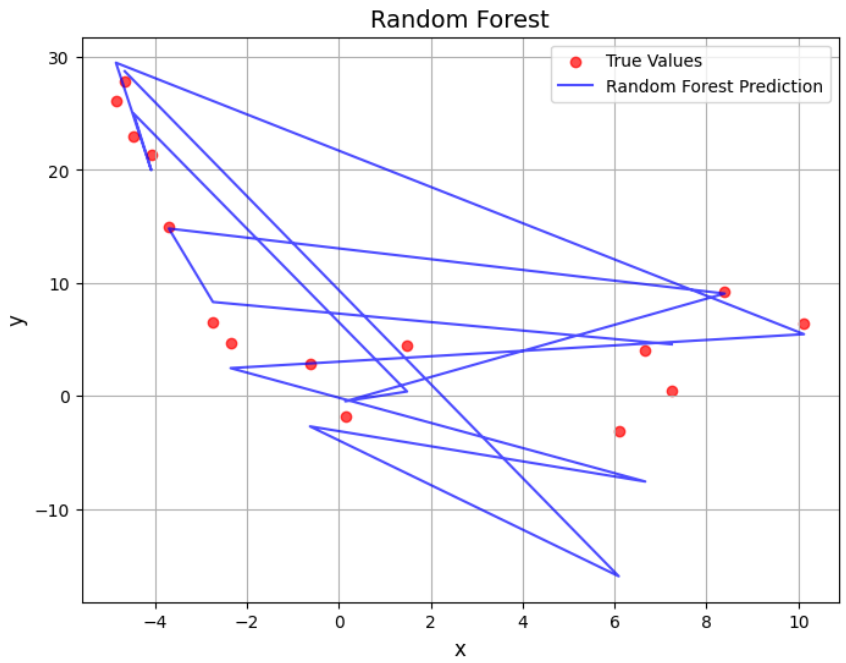
- n_estimators: تعداد درخت‌ها در جنگل.
- max_depth: حداکثر عمق هر درخت (کنترل Overfitting).
- max_features: تعداد ویژگی‌هایی که در هر تقسیم استفاده می‌شود.

مزایا:

- مقاومت در برابر Overfitting به دلیل استفاده از میانگین پیش‌بینی‌ها.
- کارایی بالا روی داده‌های غیرخطی و پیچیده.
- مقاوم در برابر نویز و داده‌های پرت.

معایب:

- ممکن است زمان اجرا و مصرف حافظه بالا باشد.
- در صورت تعداد زیاد درخت‌ها، تفسیرپذیری کم می‌شود.



تصویر 27 – مدل آموزش دیده با الگوریتم جنگل تصادفی

3. Support Vector Regressor (SVR): برای رگرسیون از نسخه‌ای از الگوریتم خود استفاده می‌کند که یک خط (یا ابرصفحه در فضای چند بعدی) پیدا می‌کند که بیشترین تعداد نقاط داده در یک بازه مشخص (ϵ) را پوشش دهد. داده‌هایی که خارج از این بازه (ϵ) قرار دارند به عنوان خطا در نظر گرفته می‌شوند و مقدار خطای آن‌ها در تابع هزینه محاسبه می‌شود.

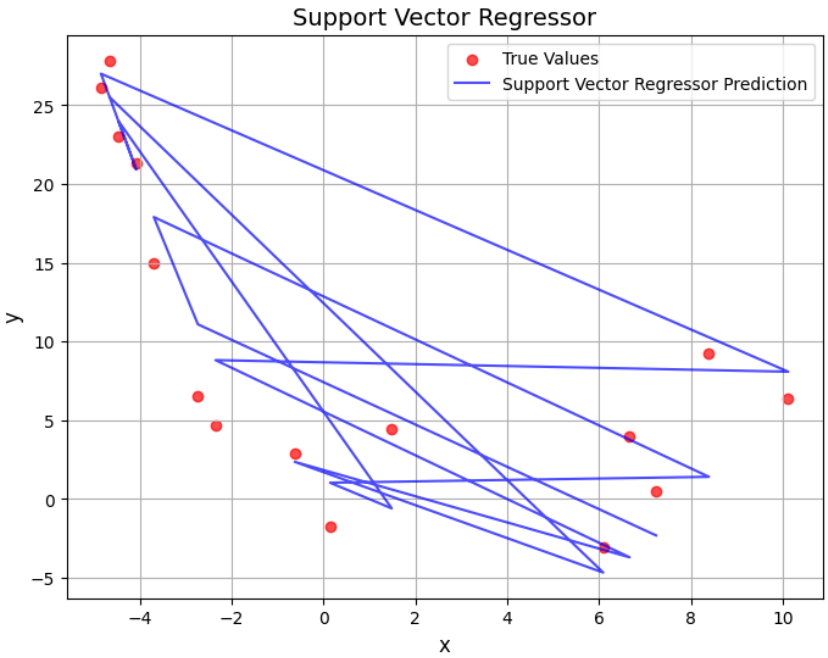
- ϵ: تعیین می‌کند که چه مقدار خطا (فاصله از خط پیش‌بینی) قابل قبول است.
- C (پارامتر هزینه): تنظیم می‌کند که مدل چقدر نسبت به داده‌های خارج از بازه حساس باشد (مقدار زیاد باعث افزایش دقت روی داده‌های آموزش و احتمال Overfitting می‌شود).
- هسته (Kernel): برای رگرسیون غیرخطی، می‌توان از هسته‌هایی مانند RBF (Radial Basis Function) استفاده کرد تا داده‌ها به فضای ویژگی بالاتر نگاشت شوند.

مزایا:

- مناسب برای داده‌های خطی و غیرخطی (با استفاده از هسته‌ها).
- مقاومت در برابر Overfitting (در صورت انتخاب پارامترهای مناسب).

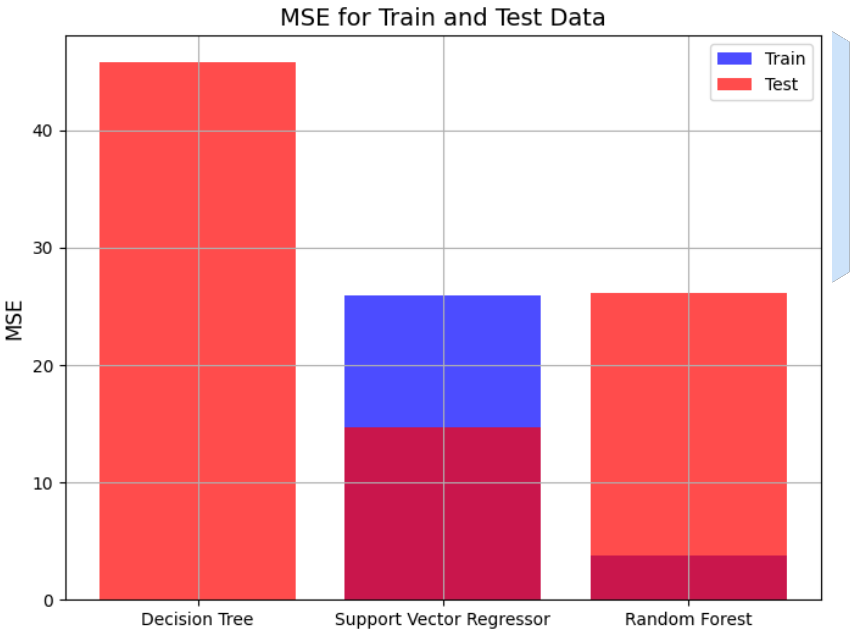
معایب:

- زمان اجرا در داده‌های بزرگ زیاد است (زیرا بر اساس نقاط پشتیبان عمل می‌کند).
- نیازمند تنظیم دقیق پارامترها (ϵ، C، نوع هسته) است.



تصویر 28 - مدل آموزش دیده با الگوریتم ساپورت وکتور

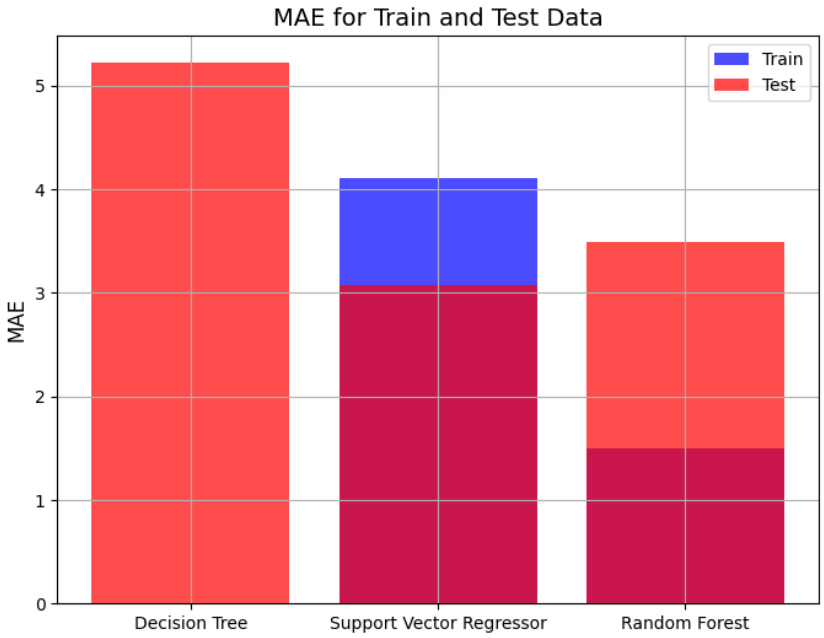
مقایسه 3 الگوریتم استفاده شده با معیار های از پیش تعیین شده:



تصویر 29 - MSE سه الگوریتم متفاوت بر روی داده های آموزش و آزمون

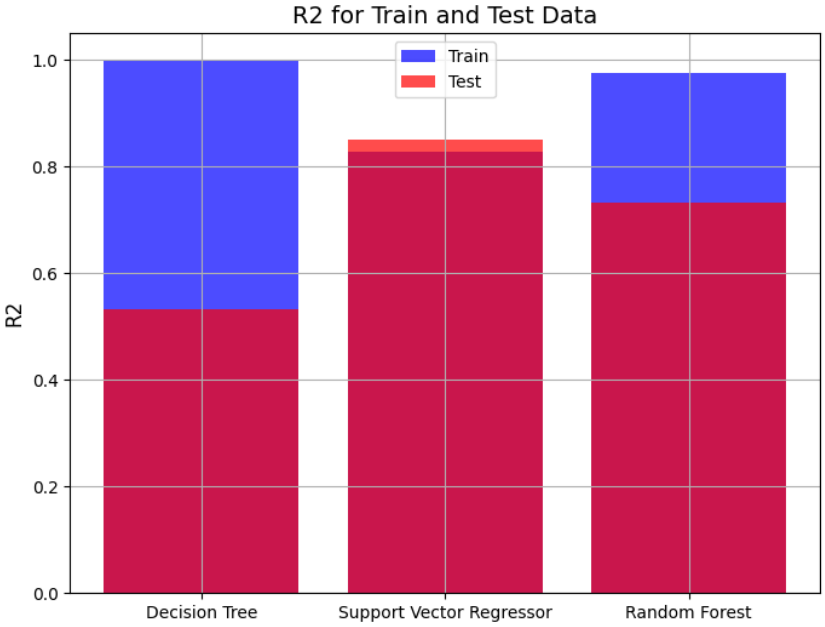
در ساپورت وکتور MSE داده های آموزش بیشتر از آزمون هستند در حالی که در جنگل تصادفی و درخت تصمیم گیری تقریبا هیچ خطایی در داده های آموزش نداریم. برتری ساپورت وکتور در داده های

آزمون هست که کمترین MSE مشاهده میشود و بعد از اون جنگل تصادفی عملکرد بهتری دارد و درخت تصمیم گیری اصلا عملکرد خوبی ندارد.



تصویر 30 – MAE سه الگوریتم متفاوت بر روی داده های آموزش و آزمون

در معیار MAE عملکرد بر روی داده های آموزش دقیقا مثل قسمت قبل شده است. در داده های آزمون این دفعه جنگل تصادفی عملکرد بهتری از بقیه الگوریتم ها دارد و پس از آن ساپورت وکتور بهتر عمل کرده است.



تصویر 31 - R-squared سه الگوریتم متفاوت بر روی داده های آموزش و آزمون

نتایج برای داده های آموزش دقیقا مثل دو قسمت قبل شده است و نتایج در قسمت آزمون مانند قسمت MSE شده است.

امتیازی:

برای اضافه کردن رگولاریزاسیون به این مدل، می‌توانید از یک پارامتر به نام λ (که به عنوان پارامتر منظم‌سازی شناخته می‌شود) استفاده کنید. این پارامتر به ماتریس (XX^T) اضافه می‌شود تا از overfitting جلوگیری شود. روش اصلی استفاده از رگولاریزاسیون برای مدل‌های خطی، Ridge Regression است که به صورت زیر فرمول‌بندی می‌شود:

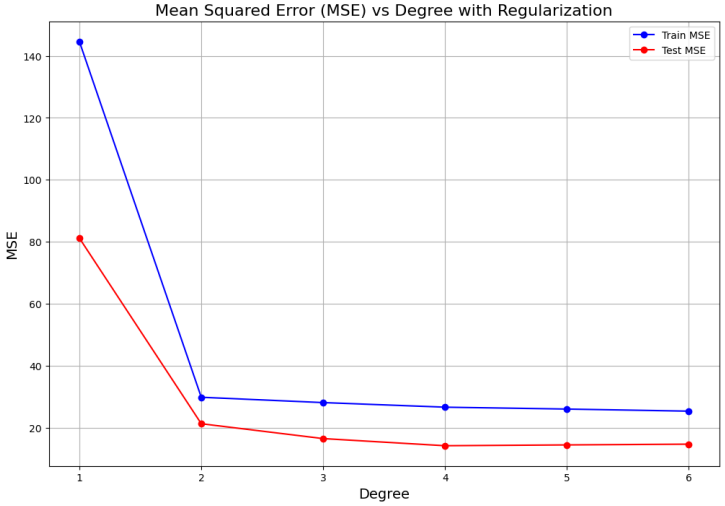
$$w = (\lambda I + XX^T)^{-1} Y^T X$$

مقدار λ پارامتر منظم‌سازی قابل تغییر است و می‌تواند روی عملکرد مدل تأثیر بگذارد. مقدار مناسب برای λ (پارامتر رگولاریزاسیون) بستگی به داده‌ها و مسئله دارد. به طور کلی، انتخاب مقدار مناسب λ فرآیندی تکراری است که با استفاده از روش‌هایی مانند cross-validation انجام می‌شود.

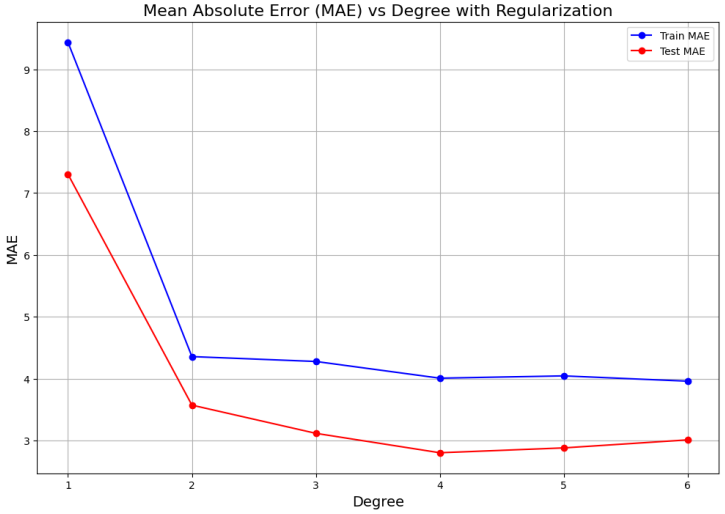
اگر λ خیلی کوچک باشد (مثلاً نزدیک به صفر)، مدل به حالت عادی (بدون رگولاریزاسیون) نزدیک می‌شود. این حالت ممکن است باعث overfitting شود، مخصوصاً برای مدل‌هایی با درجه‌های بالا.

اگر λ خیلی بزرگ باشد، مدل بیش از حد ساده می‌شود (bias بالا پیدا می‌کند) و نمی‌تواند داده‌ها را به خوبی پیش‌بینی کند (underfitting).

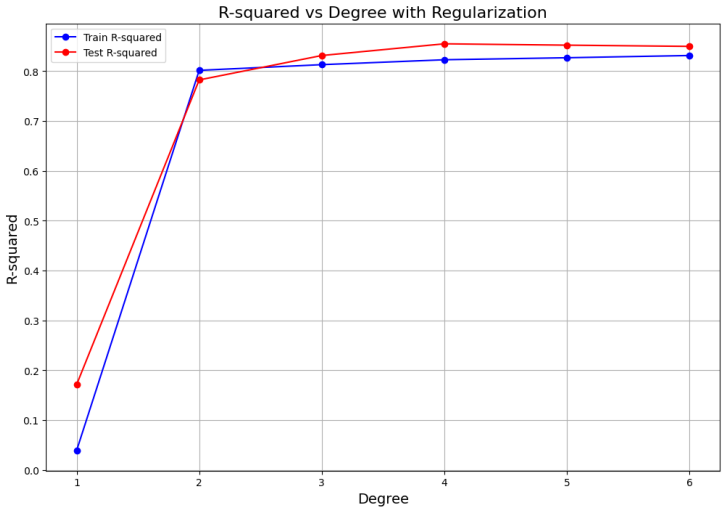
برای داده‌های نرمال‌شده (مانند اینجا)، مقادیر اولیه در بازه 0.001 تا 1000 به طور معمول آزمایش می‌شوند. اگر داده‌ها بسیار نویزی باشند، مقدار λ بزرگ‌تر به کاهش نویز و ایجاد مدل ساده‌تر کمک می‌کند. اگر داده‌ها پیچیده باشند و نیاز به مدل پیچیده‌تری داشته باشید، مقدار λ کوچک‌تر بهتر عمل می‌کند.



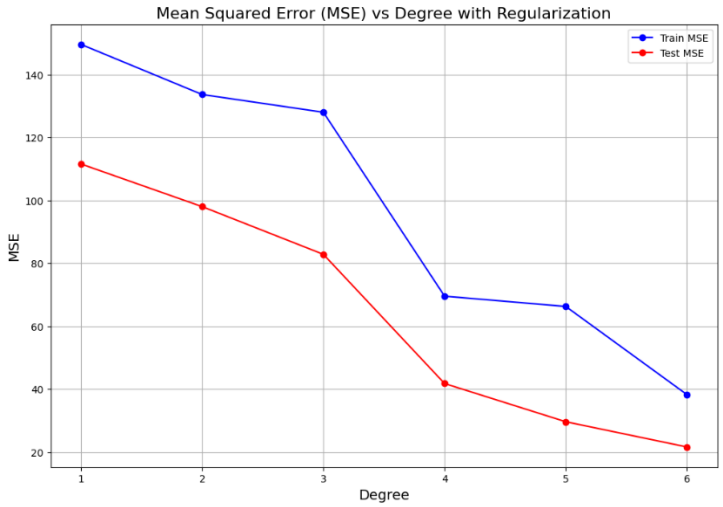
تصویر 32 - MSE مدل به ازای $\lambda=1$ و افزایش توان چندجمله ای



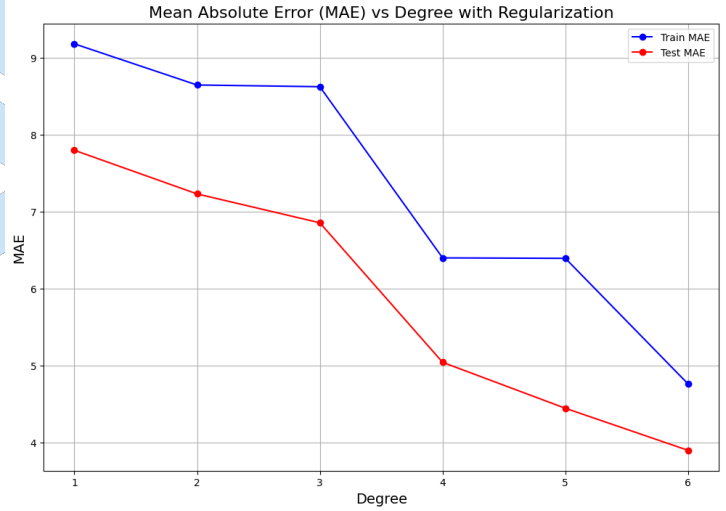
تصویر 33 - MAE مدل به ازای $\lambda=1$ و افزایش توان چندجمله ای



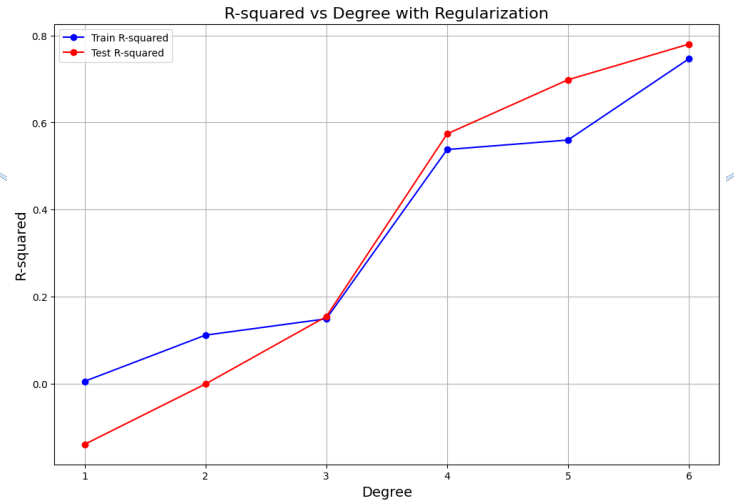
تصویر 34 - R-squared مدل به ازای $\lambda=1$ و افزایش توان چندجمله ای



تصویر 35 - MSE مدل به ازای λ=1000 و افزایش توان چندجمله ای



تصویر 36 - MAE مدل به ازای λ=1000 و افزایش توان چندجمله ای



تصویر 37 - R-squared مدل به ازای λ=1000 و افزایش توان چندجمله ای