به نام خدا

# پروژه دوم درس هوش مصنوعی

## سوال دوم

محمد امین سلطانی چم حیدری

۸۱۰۶۰۱۰۸۱

استاد مربوطه:

دکتر مسعود شریعت پناهی

بهار۱۴۰۲

**بخش دوم: پیش‌بینی عمر مفید مواد دی‌الکتریک**

مواد دی‌الکتریک به دلیل توانایی ذخیره‌ی بار الکتریکی (مانند عملکرد خازن‌ها) بصورت گسترده در صنعت مورد استفاده قرار می‌گیرند. از جمله بررسی‌هایی که در مورد این مواد صورت می‌گیرد، تعیین حداکثر ولتاژ قابل اعمال به آن‌ها در دمای کاری مشخص و برای عمر مفید مشخص است.

دادگانی که در این بخش مورد استفاده قرار می‌گیرد (فایل Performance-Degradation Data Nelson.xlsx) شامل نتایج ۱۲۸ آزمایش تعیین ولتاژ بیشینه است. هر نمونه شامل دو ویژگیِ عمر مفید (بر حسب هفته) و دمای کاری (بر حسب درجه سلسیوس) در ستون‌های اول و دوم و یک خروجیِ ولتاژ بیشینهٔ مجاز دی‌الکتریک (بر حسب کیلوولت) در ستون سوم است.

## required libraries

In [40]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pylab as pl
from sklearn import preprocessing
```

به منظور تربیت مدل برای این سوال ابتدا کتابخانه های مورد نظر در Jupyter notebook فراخوانی شده اند.

matplotlib ← نمایش data و رسم نمودار

pandas ← کار با داده های ساختار یافته مانند جداول،فایل csv و excel و...

Numpy ← انجام عملیات ریاضی،آماری و عددی

Pylab ← تجسم داده های علمی و عملی.ترکیبی از matplotlib و numpy می باشد

Sickit_learn ← یادگیری ماشین و تحلیل داده

**open file**

```
In [41]: D_electric_df = pd.read_csv("Performance-Degradation Data Nelson.csv")
         D_electric_df.head()
```

Out[41]:

| | x1 | x2 | y |
|---|---|---|---|
| 0 | 1.0 | 180.0 | 15.0 |
| 1 | 1.0 | 180.0 | 17.0 |
| 2 | 1.0 | 180.0 | 15.5 |
| 3 | 1.0 | 180.0 | 16.5 |
| 4 | 1.0 | 225.0 | 15.5 |

```
In [42]: D_electric_df.dtypes
```

```
Out[42]: x1     float64
         x2     float64
         y      float64
         dtype: object
```

در مرحله بعد فایل های داده های مربوط به عمر مفید، دمای کاری و خروجی ولتاژ بیشینه مجاز دی الکتریک به کمک دستور pd از کتابخانه pandas باز شده اند.

**defin x & y**

```
In [43]: feature_df = D_electric_df[['x1', 'x2']]
         X = np.asarray(feature_df)
         X[0:5]
```

```
Out[43]: array([[  1., 180.],
                [  1., 180.],
                [  1., 180.],
                [  1., 180.],
                [  1., 225.]])
```

```
In [44]: D_electric_df['y'] = D_electric_df['y'].astype('float')
         y = np.asarray(D_electric_df['y'])
         y [0:20]
```

```
Out[44]: array([15. , 17. , 15.5, 16.5, 15.5, 15. , 16. , 14.5, 15. , 14.5, 12.5,
                11. , 14. , 13. , 14. , 11.5, 14. , 16. , 13. , 13.5])
```

در این قسمت آرایه های مربوط به دمای کاری و عمر مفید به عنوان متغیرهای x1 و x2 و داده های مربوط به خروجی ولتاژ بیشینه مجاز دی الکتریک به عنوان خروجی درمتغیر y تعریف می شوند. با توجه به اینکه کتابخانه pandas آرایه ها را به صورت str فراخوانی می کند و سوال نیاز به داده های ریاضی دارد در این قسمت برای تعریف آرایه های ورودی و خروجی در متغیرهای x , و y از دستور numpy استفاده می شود.

**normalizing x**

In [45]:
```
print(X[0:10])
Scaler= preprocessing.StandardScaler().fit(X)
X =Scaler.transform(X.astype(float))
print(X[0:10])
```

```
[[  1. 180.]
 [  1. 180.]
 [  1. 180.]
 [  1. 180.]
 [  1. 225.]
 [  1. 225.]
 [  1. 225.]
 [  1. 225.]
 [  1. 250.]
 [  1. 250.]]
[[-0.94101309 -1.49618805]
 [-0.94101309 -1.49618805]
 [-0.94101309 -1.49618805]
 [-0.94101309 -1.49618805]
 [-0.94101309 -0.21374115]
 [-0.94101309 -0.21374115]
 [-0.94101309 -0.21374115]
 [-0.94101309 -0.21374115]
 [-0.94101309  0.49872935]
 [-0.94101309  0.49872935]]
```

سپس با دستور بالا داده های ورودی normalize می شوند.علی رغم اینکه درصورت سوال مانن سوال یک ذکر نشده که داده ها normalize شوند ولی برای جلوگیری از بیش پردازش مدل در این قسمت نیز داده های ورودی normalize شده اند.

**Data separation**

In [46]:
```
X_train, X_test, y_train, y_test =train_test_split( X, y, test_size=0.25, random_state=50)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (96, 2) (96,)
Test set: (32, 2) (32,)
```

در صورت سوال ذکر شده که 75 درصد از داده ها برای تمرین و 25 درصد از داده ها برای تست در نظر گرفته شوند.برای این کار می توان از کتابخانه scikit learn استفاده کرد یا به صورتی که در کد بالا مشاهده می شود داده ها را به دوگروه تمرین وتست تقسیم کرد.

همچنین به منظور اینکه نتایج گزارش شده در این فایل با نتایج بدست آمده از کد همواره برابر باشد حالت random state به صورت ثابت تعریف می شود.می توان برای تصادفی بدست آمدن داده ها حالت random state را پاک کرد که در این صورت با هربار run کردن کد باتوجه به اینکه داده ها تصادفی در نظر گرفته می شوند نتایج متفاوت خواهد بود.

الف) به کمک نرم افزار پایتون و کتابخانه‌های مناسب، داده‌های فایل را وارد و به کمک دستور مناسب از کتابخانه‌ی sklearn، رگرسیون را با کرنل‌های خطی، RBF، چندجمله‌ای درجه ۲ و سیگموئیدی انجام داده و خطای مطلق میانگین ( mean absolute error) و امتیاز R2 (R2-score) را در هر کدام به کمک روش k-fold cross validation (با k=4 یعنی در هر حالت ۲۵٪ از داده‌ها با انتخاب تصادفی پیش فرض sklearn برای آزمایش در نظر گرفته شود) به دست آورده و عملکرد چهار تابع کرنل (میانگین امتیاز به دست آمده برای داده‌های آزمایش) را مقایسه نمایید.

**در این روش داده ها بصورت تصادفی به k گروه تقسیم می شوند و مدل به تعداد k بار تربیت و اعتبارسنجی می شود. در هر بار، یکی از گروه ها به عنوان داده ی تست کنار گذاشته می شود و مدل به کمک K-1 گروه دیگر تربیت شده و سپس با گروه کنار گذاشته شده اعتبارسنجی می شود.**

**calculate R2-Score and Mean absolte error with K-Fold cross validation**

In [47]:
```
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
import numpy as np

model_linear = SVR(kernel='linear')
model_rbf = SVR(kernel='rbf')
model_poly = SVR(kernel='poly')
model_sigmoid = SVR(kernel='sigmoid')

mae_linear = -1 * cross_val_score(model_linear, X, y, cv=4, scoring='neg_mean_absolute_error')
mae_rbf = -1 * cross_val_score(model_rbf, X, y, cv=4, scoring='neg_mean_absolute_error')
mae_poly = -1 * cross_val_score(model_poly, X, y, cv=4, scoring='neg_mean_absolute_error')
mae_sigmoid = -1 * cross_val_score(model_sigmoid, X, y, cv=4, scoring='neg_mean_absolute_error')

r2_linear = cross_val_score(model_linear, X, y, cv=4, scoring='r2')
r2_rbf = cross_val_score(model_rbf, X, y, cv=4, scoring='r2')
r2_poly = cross_val_score(model_poly, X, y, cv=4, scoring='r2')
r2_sigmoid = cross_val_score(model_sigmoid, X, y, cv=4, scoring='r2')

print("MAE (linear kernel):", np.mean(mae_linear))
print("MAE (RBF kernel):", np.mean(mae_rbf))
print("MAE (polynomial kernel):", np.mean(mae_poly))
print("MAE (sigmoid kernel):", np.mean(mae_sigmoid))
print("R2 score (linear kernel):", np.mean(r2_linear))
print("R2 score (RBF kernel):", np.mean(r2_rbf))
print("R2 score (polynomial kernel):", np.mean(r2_poly))
print("R2 score (sigmoid kernel):", np.mean(r2_sigmoid))
```

باتوجه به صورت سوال برای بدست آوردن MAE و R2-Score با استفاده از کرنل های ذکر شده مدل تربیت شده از نوع SVM می باشد.

لذا از کتابخانه scikit learn دستورSVR برای تعریف مدل، دستور mean absolute error برای محاسبه خطا، دستور R2-score برای محاسبه امتیاز r2 و دستور cross_val_score فراخوانی می شوند.

سپس مدل مربوطه با linear kernel , RBF kernel , polnomial kernel , sigmoid kernel تعریف می شود و و موارد MAE و R2-Score برای هر کدام از کرنل های ذکر شده آن محاسبه می شوند.

```
MAE (linear kernel): 2.0696381245514797
MAE (RBF kernel): 2.2985670505536637
MAE (polynomial kernel): 3.762904507074092
MAE (sigmoid kernel): 3.00050725303824
R2 score (linear kernel): 0.07312273727223464
R2 score (RBF kernel): 0.10895320219817875
R2 score (polynomial kernel): -1.2982322540950209
R2 score (sigmoid kernel): -0.157566767233079
```

MAE میانگین خطای مطلق برای نتایج بدست آمده از مدل تربیت شده نسبت به مقادیر واقعی می باشد.

R2-Score یک معیار ارزیابی برای سنجش مدل تربیت شده به کمک روش های متفاوت ماشین لرنینگ می باشد و بیانگر میزان تطابق بین نتایج پیش بینی شده توسط مدل با داده های واقعی می باشد که مقدار آن بین صفر و یک است.مقدار یک به معنای تطابق کامل نتایج بدست آمده از مدل با داده های واقعی می باشد و مقدار صفر به معنای عدم تطابق آن ها است.

همچنین بدست آمدن مقدار منفی برای R2-Score یعنی مدل تربیت شده از یک مدل تصادفی ضعیف تر عمل می کند.

در این قسمت از سوال که به کمک روش k-fold cross validation مدل تربیت شد برای هر کرنل نتایج متفاوتی بدست آمده است.

در بین کرنل های تعریف شده کمترین MAE برای کرنل خطی و بیشترین آن برای کرنل چندجمله ای می باشد لذا با توجه به این معیار مدل linear kernel بهترین مدل تربیت شده است. ولی از لحاظ سنجش R2-Score علی رغم اینکه مقادیر این معیار کوچکتر از حد توقع است ولی در بین آن ها بیشترین امتیاز را RBF kernel دارد و میتوان گفت از نظر معیار R2-Score بهترین مدل RBF kernel می باشد.

همچنین مدل های تربیت شده با polynomial kernel و sigmoid kernel دارای R2-score منفی هستند یعنی از مدل های تصادفی ضعیف تر عمل می کنند و به طور کلی مدل polynomial kernel باتوجه به خطای زیاد و R2-score منفی ضعیف ترین مدل در بین مدل های تربیت شده می باشد.

ب) خواسته‌ی مورد الف را به کمک L2-regularization با پارامتر alpha=1, 2 (دو مقدار) برای توابع کرنل بخش الف (چهار تابع) به دست آورده و نتایج را مقایسه نمایید. با توجه به تغییرات به وجود آمده در دقت رگرسیون درداده‌های آزمایش نسبت به بخش قبل، چه نتایجی می‌توان گرفت؟

**Regularization یا عادی سازی یعنی محدود نگه داشتن اندازه ضرایب جملات به کمک افزودن یک جمله جدید برای جلوگیری از بیش پردازش. در Regularization با تغییر درجه alpha می توان دقت را کمو زیاد کرد. با انتخاب مقدار بهینه برای alpha می توان تعیین کرد چه مقدار هدف بالا بردن دقت می باشد و چه مقدار جلوگیری از overfit مد نظر استطبیعی است که هرچه دقت افزایش یابد خطر over fit بیشتر مدل را تهدید می کند.**

**calculate R2-Score and Mean absolte error L2 Regularization with alpha=1**

In [30]:

```
from sklearn.kernel_ridge import KernelRidge
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

model_linear = KernelRidge(kernel='linear', alpha=1)
model_rbf = KernelRidge(kernel='rbf', alpha=1)
model_poly = KernelRidge(kernel='polynomial', alpha=1)
model_sigmoid = KernelRidge(kernel='sigmoid', alpha=1)

model_linear.fit(X_train, y_train)
model_rbf.fit(X_train, y_train)
model_poly.fit(X_train, y_train)
model_sigmoid.fit(X_train, y_train)

yhat_linear = model_linear.predict(X_test)
yhat_rbf = model_rbf.predict(X_test)
yhat_poly = model_poly.predict(X_test)
yhat_sigmoid = model_sigmoid.predict(X_test)

mae_linear = mean_absolute_error(y_test, yhat_linear)
mae_rbf = mean_absolute_error(y_test, yhat_rbf)
mae_poly = mean_absolute_error(y_test, yhat_poly)
mae_sigmoid = mean_absolute_error(y_test, yhat_sigmoid)

r2_linear = r2_score(y_test, yhat_linear)
r2_rbf = r2_score(y_test, yhat_rbf)
r2_poly = r2_score(y_test, yhat_poly)
r2_sigmoid = r2_score(y_test, yhat_sigmoid)


print("MAE (linear kernel with regularization):", mae_linear)
print("MAE (RBF kernel with regularization):", mae_rbf)
print("MAE (polynomial kernel with regularization):", mae_poly)
print("MAE (sigmoid kernel with regularization):", mae_sigmoid)
print("R2 score (linear kernel with regularization):", r2_linear)
print("R2 score (RBF kernel with regularization):", r2_rbf)
print("R2 score (polynomial kernel with regularization):", r2_poly)
print("R2 score (sigmoid kernel with regularization):", r2_sigmoid)
```

در این قسمت مدل با کرنل های گفته شده به کمک Regularization با alpha=1 و alpha=2 تربیت شده است.

برای تربیت مدل به این روش از کتابخانه scikit learn تابع kernel ridge فراخوانی شده است. در این کد ابتدا مدل ها با هریک از توابع کرنل ذکر شده تعریف شده اند.سپس مانند قسمت قبل MAE و R2-Score تعریف شده و برای هر مدل محاسبه می شود. در این قسمت چون از روش k-fold cross validation کمک گرفته نشده مقدار خروجی پیشبینی شده تو سط تابع نیز تعریف می شود تا با مقایسه آن با مقادیر واقعی MAE و R2-Score محاسبه شوند.

```
MAE (linear kernel with regularization): 11.29929843954404
MAE (RBF kernel with regularization): 1.1016312676690652
MAE (polynomial kernel with regularization): 1.021410574790 98
MAE (sigmoid kernel with regularization): 1.2015347058745174
R2 score (linear kernel with regularization): -8.033753233858183
R2 score (RBF kernel with regularization): 0.8540177945904703
R2 score (polynomial kernel with regularization): 0.8740271562889432
R2 score (sigmoid kernel with regularization): 0.8564279597644999
```

همانگونه که مشاهده می شود که با استفاده از Regularization مقادیر MAE و R2-Score تغییر کرده اند.

در قسمت linear kernel مقدار خطا افزایش یافته که نتیجه گرفته می شود که برای کرنل خطی Regularization مقدار MAE را افزایش می دهد که نامطلوب است.همچنین برای این kernel مقدار R2-Score کاهش یافته و منفی شده که نشان می دهد این مدل از مدل تصادفی ضعیف تر عمل می کند.

ولی برای سایر کرنل ها یعنی RBF kernel , polynomial kernel , sigmoid kernel مشاهده می شود که مقدار MAE کاهش یافته که این موضوع در تربیت مدل ما مطلوب می باشد.

همچنین R2-Score برای این کرنل ها افزایش یافته که این موضوع نیز مطلوب می باشد.

لذا می توان گفت استفاده از Regularization در RBF kernel , polynomial kernel , sigmoid kernel موجب افزایش R2-Score و کاهش MAE می شود ولی در linear kernel موجب کاهش R2-

Score وافزایش MAE می شود پس به طور کلی Regularization در کرنل خطی موجب تضعیف مدل و در سایر کرنل ها موجب بهبود مدل شده است.

مشاهده می شود که polynomial kernel که در قسمت قبل ضعیف ترین مدل برای پیشبینی بود با استفاده از Regularization تبدیل به بهترین مدل با کمترین MAE و بیشترین R2-Score شده است.

```
MAE  (linear kernel with regularization): 11.3030567335848
MAE  (RBF kernel with regularization): 1.2911341613149134
MAE  (polynomial kernel with regularization): 1.0801986667755576
MAE  (sigmoid kernel with regularization): 2.2035119653940543
R2 score (linear kernel with regularization): -8.039870553817783
R2 score (RBF kernel with regularization): 0.8005220564348796
R2 score (polynomial kernel with regularization): 0.8642112090064145
R2 score (sigmoid kernel with regularization): 0.4784939846338221
```

نتایج بالا برای استفاده از Regularization با alpha =2 می باشد.نتایج این جدول در مقایسه با مدل alpha =1 و مدل k-fold cross validation در جدول زیر آورده شده است.

| | MAE k-fold | R2-Score k-fold | MAE Alpha=1 | R2-Score Alpha=1 | MAE Alpha=2 | R2-Score Alpha=2 |
|---|---|---|---|---|---|---|
| **linear** | 2.06963 | 0.07312 | 11.29929 | -8.03375 | 11.30305 | -8.03987 |
| **RBF** | 2.29856 | 0.10895 | 1.10163 | 0.85401 | 1.29113 | 0.80052 |
| **poly** | 3.76290 | -1.29823 | 1.02141 | 0.87402 | 1.08019 | 0.86421 |
| **sigmoid** | 3.00050 | -0.15756 | 1.20153 | 0.85642 | 2.20351 | 0.47849 |

در جدول بالا ستون دوم و سوم از سمت چپ مربوط به مدل تعریف شده با k-fold cross validation ،ستون سوم و چهارم مربوط به مدل تعریف شده با Regularization و alpha=1 و ستون پنجم و ششم نشان دهنده نتایج حاص از تعریف مدل با Regularization و alpha=2 می باشد.

نتایج حاصل:

در مدل k-fold cross validation تابع کرنل خطی دارای کمترین MAE و تابع کرنل چندجمله ای دارای بیشترین MAE می باشد.

در این مدل کرنل RBF بیشترین R2-Score و کرنل چندجمله ای کمترین R2-Score را دارد. همچنین در این مدل R2-Score برای کرنل چند جمله ای و سیگموییدی منفی می باشد یعنی این مدل با این کرنل ها از مدل تصادفی ضعیف تر عمل میکند.

در مدل بعدی و استفاده از Regularization با alpha=1 مشاهده می شود که در کرنل خطی مقدار MAE افزایش یافته و مقدار R2Score کاهش یافته است و در RBF kernel , polynomial kernel , sigmoid kernel مقدار MAE کاهش یافته و مقدار R2Score افزایش یافته است. میتوان نتیجه گرفت که استفاده از Regularization مو جب بهبود RBF kernel , polynomial kernel , sigmoid kernel و موجب تضعیف linear kernel شده است.

همچنین در قسمت آخر که Regularization با alpha =2 انجام شده است مشاهده می شود که به طور کلی مقادیر MAE افزایش یافته و مقادیر R2-Score کاهش یافته اند که این مورد نامطلوب است. البته در توابع RBF kernel , polnomial kernel , linear kernel مقدار افزایش MAE و کاهش R2-Score کم می باشد ولی در sigmoid kernel این مقادیر قابل توجه است.

پس میتوان نتیجه گرفت افزایش alpha موجب تضعیف جزئی مدل های linear kernel , polynomial kernel , RBF kernel و تضعیف قابل توجه sigmoid kernel شده است.

**به طور کلی می توان نتیجه گرفت که Regularization نسبت به k-fold cross validation "معمولا" موجب بهبود MAE و R2-Score می شود.همچنین می توان نتیجه گرفت با کاهش میزان alpha مقادیر MAE , R2-Score بهبود می یابند ولی در صورت کاهش زیاد موجب over fit شدن مدل می شود. در قسمت بعدی مقدار بهینه ای برای alpha این مدل بدست آورده شده است.**

ج) با تغییر پارامتر Regularization در مقادیر {0.2, 0.8, 1,5, 10, 20, 50, 300} به ازای توابع کرنل خطی، چندجمله‌ای درجه ۲ و ۳ و ۴ و هم چنین RBF مقادیر بهینه را به ازای هر کرنل و هم چنین بهترین کرنل را با بهترین امتیاز R2 به دست بیاورید (از دستور gridsearchcv استفاده نمایید). مقادیر نزدیک صفر برای امتیاز R2 به چه معنا هستند؟

## finding best R-2Score

```
from sklearn.kernel_ridge import KernelRidge
from sklearn.model_selection import GridSearchCV
import numpy as np

model = KernelRidge()

param_grid = {'alpha': [0.2, 0.8, 1, 5, 10, 20, 50 ,300],
              'degree': [2,3,4],
              'kernel': ['linear', 'rbf', 'polynomial']}

grid_search = GridSearchCV(model, param_grid=param_grid, cv=4, scoring='r2')

grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best R2 score: ", grid_search.best_score_)
```

برای محاسبه ی بهترین R2-Score و بهترین ضریب alpha همانگونه که در صورت سوال گفته شده از دستور grid search cv استفاده شده است. ابتدا مدل حل را با kernelridge تعریف کرده و سپس با دستور grid search cv و fit کردن آن بر روی مدل بهترین R2-Score و بهترین ضریب alpha محاسبه می شوند.

```
Best parameters:  {'alpha': 0.2, 'degree': 3, 'kernel': 'polynomial'}
Best R2 score:  0.7923436501510474
```

**همانگونه که مشاهده می شود بهترین ضریب alpha برابر با 0.2 برای تابع چند جمله ای از درجه سه بدست آمده و برای این مدل امتیاز R2 برابر با 0.7923 محاسبه شده است.**

R2-Score یک معیار ارزیابی برای سنجش مدل تربیت شده به کمک روش های متفاوت ماشین لرنینگ می باشد و بیانگر میزان تطابق بین نتایج پیش بینی شده توسط مدل با داده های واقعی می باشد که مقدار آن بین صفر و یک است. مقدار یک به معنای تطابق کامل نتایج بدست آمده از مدل با داده های واقعی می باشد و مقدار صفر به معنای عدم تطابق نتایج بدست آمده از مدل با داده های واقعی می باشد .

د) در نرم افزار متلب به کمک دستور مناسب، فایل داده‌ها (با فرمت Excel) را وارد کرده و رگرسیون غیر خطی را با تابع زیر بر روی داده‌ها اعمال و امتیاز R2 و ریشه‌ی میانگین مربعات خطا را برای تابع برازش شده (به دست آوردن ضرایب مجهول b) به دست آورید.

$$\log(y) = b_1 - b_2.x_1.\exp(-b_3.x_2)$$

```
clc;
clear all;
% Open data
data = readtable('Performance-Degradation Data Nelson.xlsx');

% Access to data
x1 = data.x1;
x2 = data.x2;
x = [x1, x2];
y = data.y;
b_initial=zeros(1,3);

% Define function and Calculation of unknown coefficients
fun = @(b, x) b(1) - b(2)*x(:,1).*exp(-b(3)*x(:,2));
[b, resnorm, ~, exitflag, output] = lsqcurvefit(fun, b_initial, x, log(y));

% Optimization of unknown coefficients and calculate R2-Score & RSME
y_fit = exp(fun(b, x));
RMSE = sqrt(resnorm/length(y));
R2 = corrcoef(y, y_fit).^2;
% Print the resault
fprintf('Coefficients: b1=%.5f, b2=%.5f, b3=%.5f\n', b(1), b(2), b(3));
fprintf('R2: %.5f\n', R2(1,2));
fprintf('RMSE: %.5f\n', RMSE);
```

برای اعمال رگرسیون خطی بر روی داده های ورودی ابتدا فایل مربوطه باز می شود. سپس داده ها در دو متغیر x و y ذخیره می شوند. همچنین یک ماتریس b initial با درایه های صفر برای قرار گیری مقادیر ضرایب تعریف می شود. سپس تابع رگرسیون مطابق خواسته سوال تعریف می شود و ضرایب مجهول برای آن محاسبه می شود. در پایان تابع تعریف شده بر روی داده ها fit شده و R2-Score و Root mean squared error برای آن محاسبه می شود.

```
Coefficients: b1=2.58141, b2=0.00000, b3=-0.04662
R2: 0.85409
RMSE: 0.18255
```

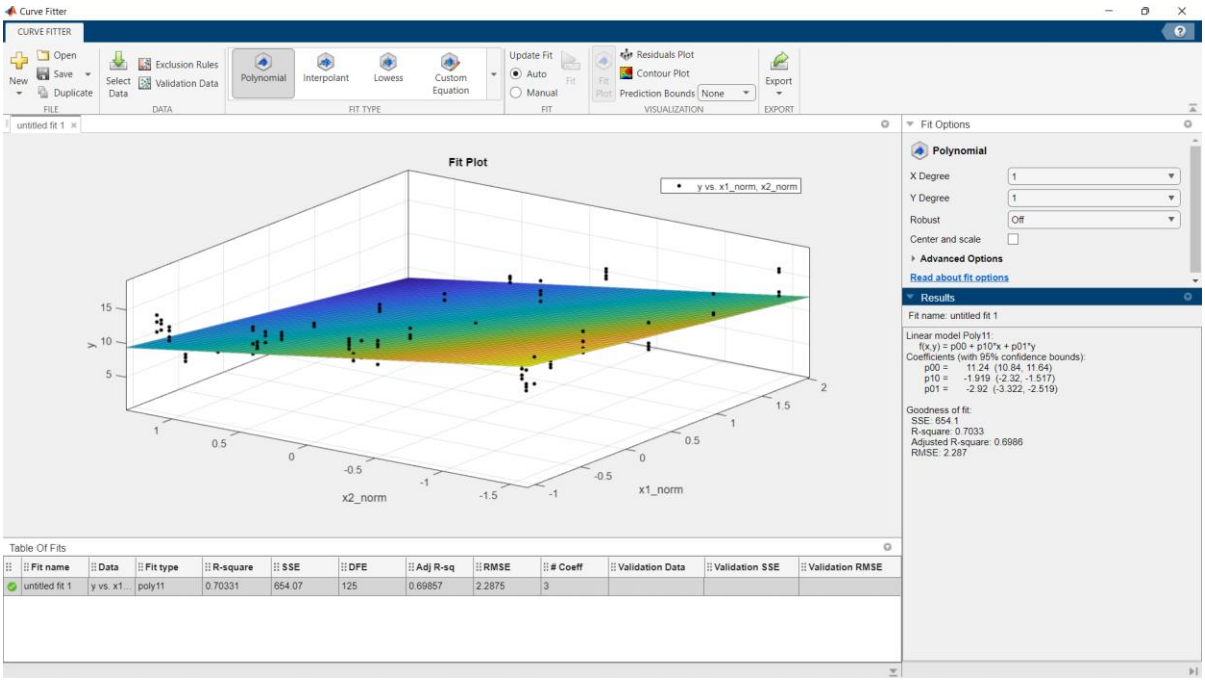
مقدار ضرایب بهینه شده b1 تا b3 ، R2-Score , RMSE مطابق شکل بالا بدست می آید.

ه) به کمک دستور cftool در متلب و با استفاده از داده‌های آزمایش، تابع چندجمله‌ای را به ازای مقادیر مختلف درجات x1 و x2 بر داده‌ها برازش کرده و در حالتی که امتیاز R2 بیشینه می‌شود، مقادیر امتیاز R2 و RMS خطا را به همراه ضرایب چندجمله‌ای و رویه‌ی ایجاد شده ارائه نمایید.

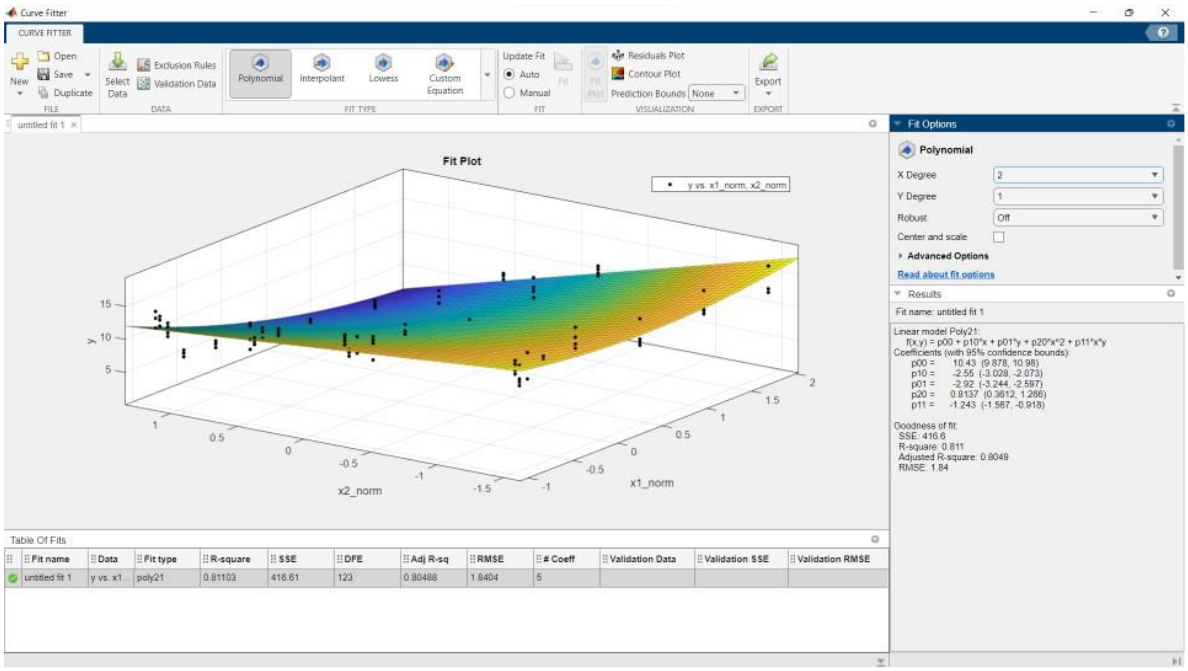
```
clc;
clear all;
% Open data
d = 'Performance-Degradation Data Nelson.xlsx';
data = readtable(d);
% Access to data
x1 = data.x1;
x2 = data.x2;
x = [x1, x2];
y = data.y;
b_initial=zeros(1,3)

x1_norm = minmax(x1);
x2_norm = minmax(x2);
x1_norm = zscore(x1);
x2_norm = zscore(x2);

% Define function and Calculation of unknown coefficients
fun = @(b, x) b(1) - b(2)*x(:,1).*exp(-b(3)*x(:,2));
[b, resnorm, ~, exitflag, output] = lsqcurvefit(fun, b_initial, x, log(y));

% Optimization of unknown coefficients and calculate R2-Score & RSME
y_fit = exp(fun(b, x));
RMSE = sqrt(resnorm/length(y));
R2 = corrcoef(y, y_fit).^2;
% Print the resault
fprintf('Coefficients: b1=%.5f, b2=%.5f, b3=%.5f\n', b(1), b(2), b(3));
fprintf('R2: %.5f\n', R2(1,2));
fprintf('RMSE: %.5f\n', RMSE);
load data.mat
```

برای برازش تابع چند جمله ای ابتدا داده های ورودی normalize می شوند. سپس برای برازش آن از دستور load datamat استفاده می شود. پس از اجرای کد در command windows عبارت cftool را اجرا کرده تا پنجره ی متلب باز شود.

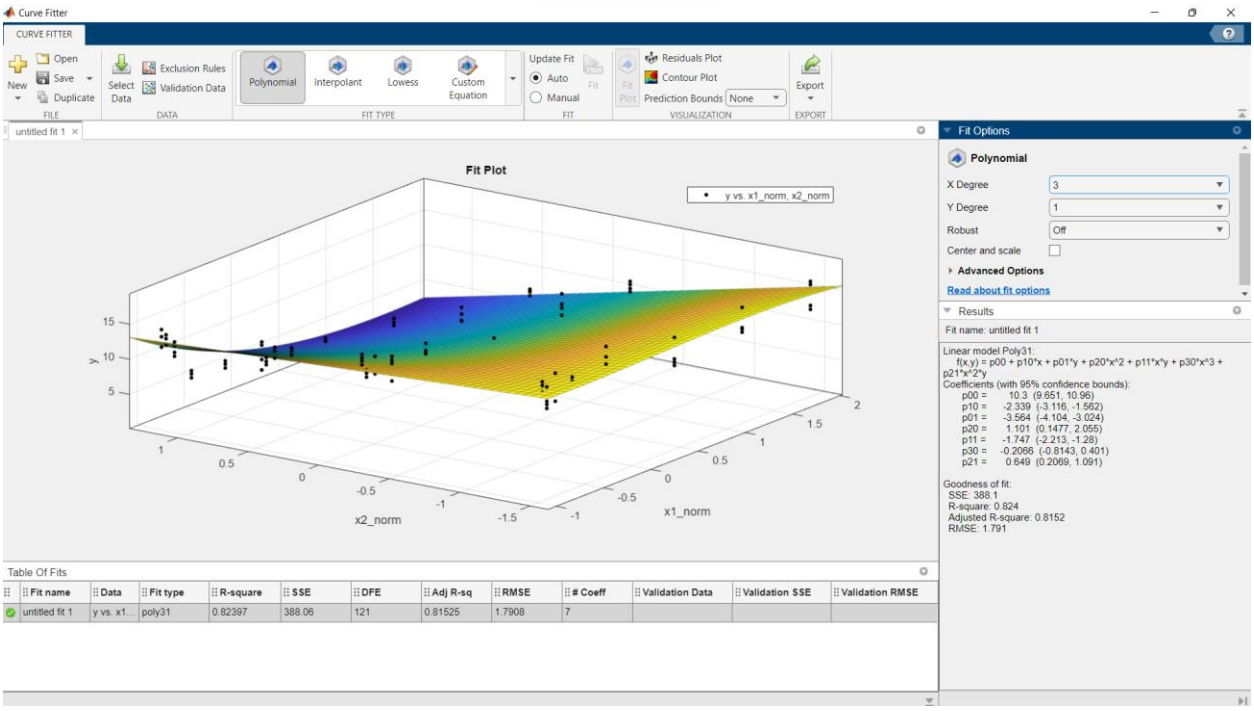
پس از باز شدن صفحه متلب مطابق شکل زیر از پنجره ی fit type گزینه polynomial انتخاب می شود تا بتوان درجات مختلف آن را با یکدیگر مقایسه کرد.



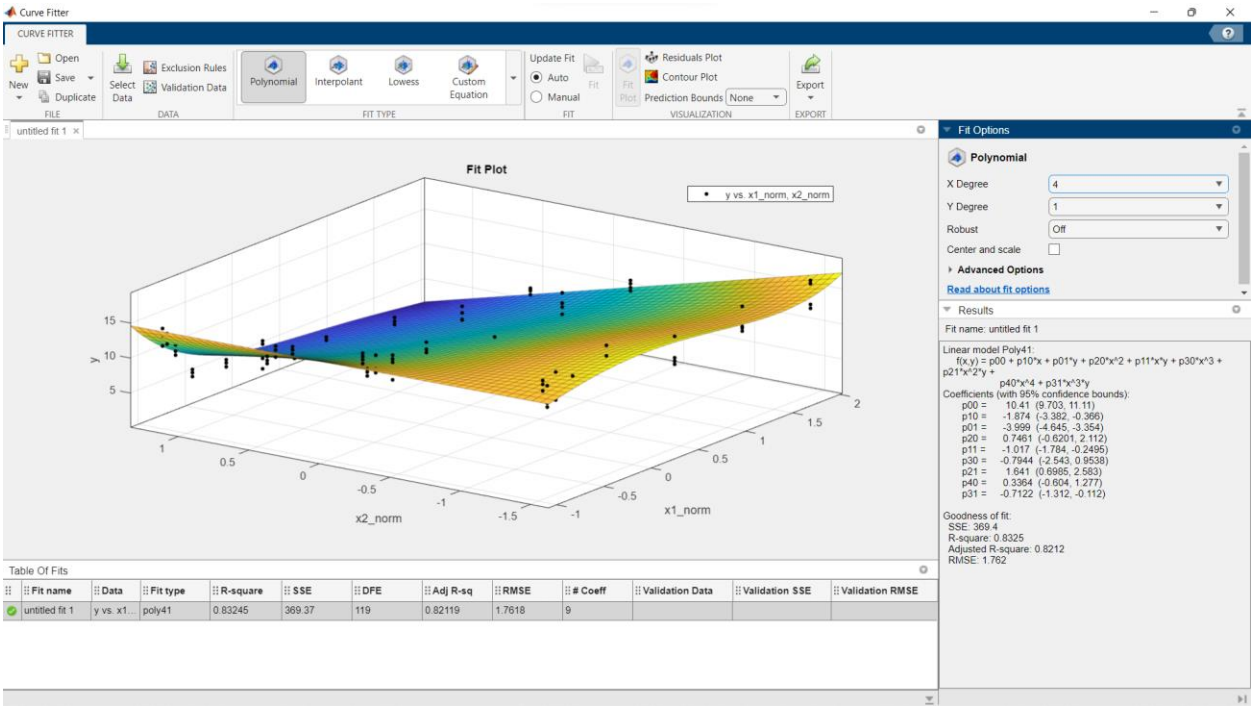

پس از این مرحله از پنجره ی fit option در سمت راست تصویر درجات آزادی ورودی های مدل قابل مشاهده هستند. در این بخش قابلیت اعمال ۱۵درجه آزادی متفاوت برای مقادیر x و y وجود دارد که برای هرکدام رویه بدست آمده در شکل های زیر آورده شده است. همچنین در پنجره ی Results مقادیر R2-Score و RMSE و ضرایب چند جمله ای برای هر رویه قابل مشاهده می باشد.



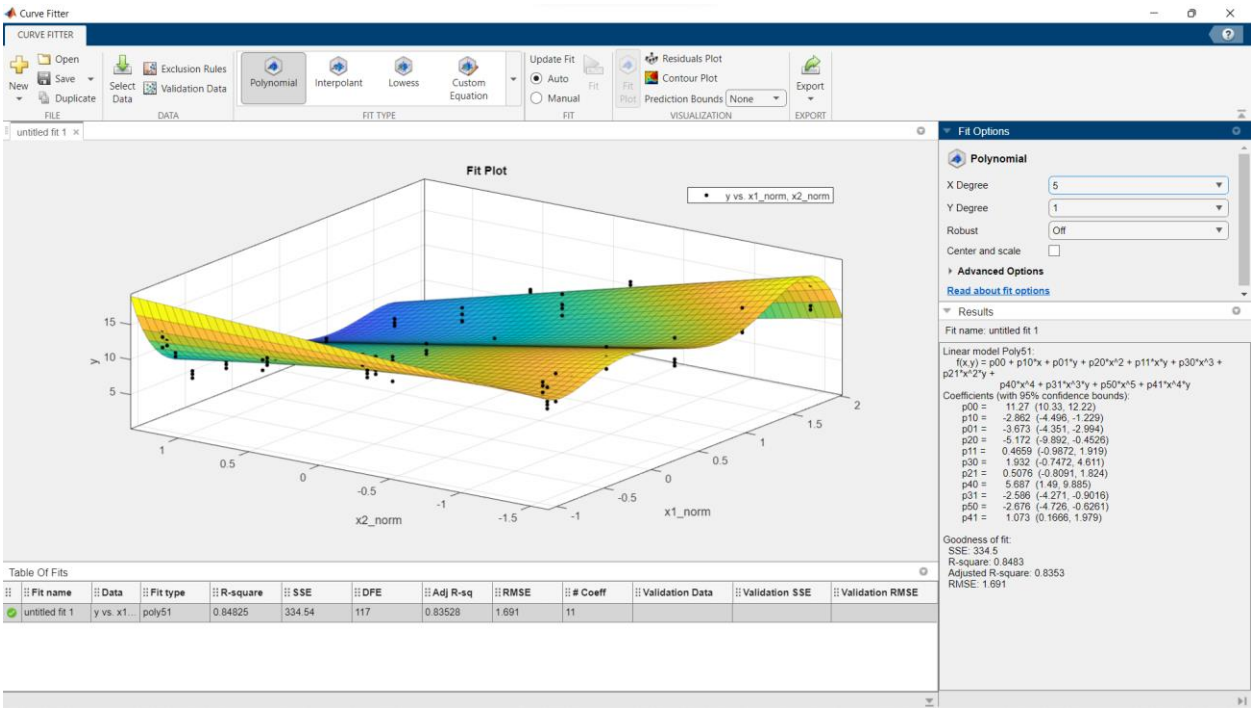

X degree = 2  Y degree=1  R2-Score=0.811  RMSE = 1.84



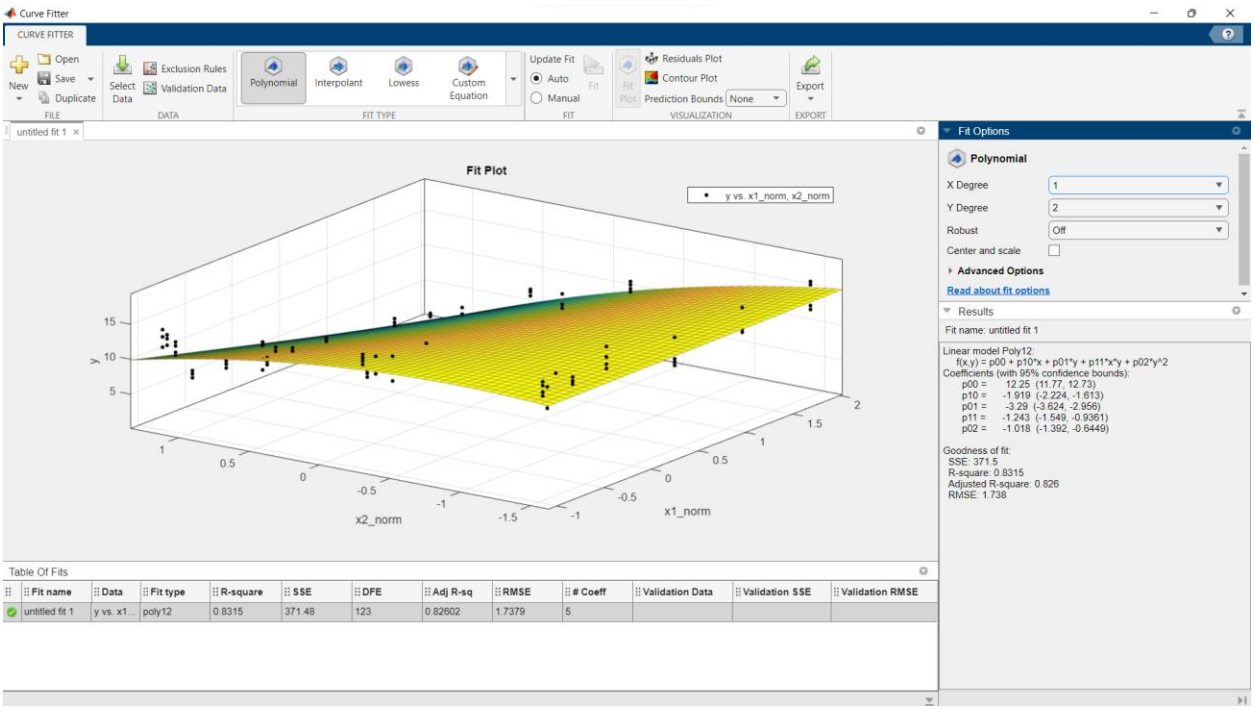

X degree = 3 Y degree=1 R2-Score=0.824 RMSE = 1.791



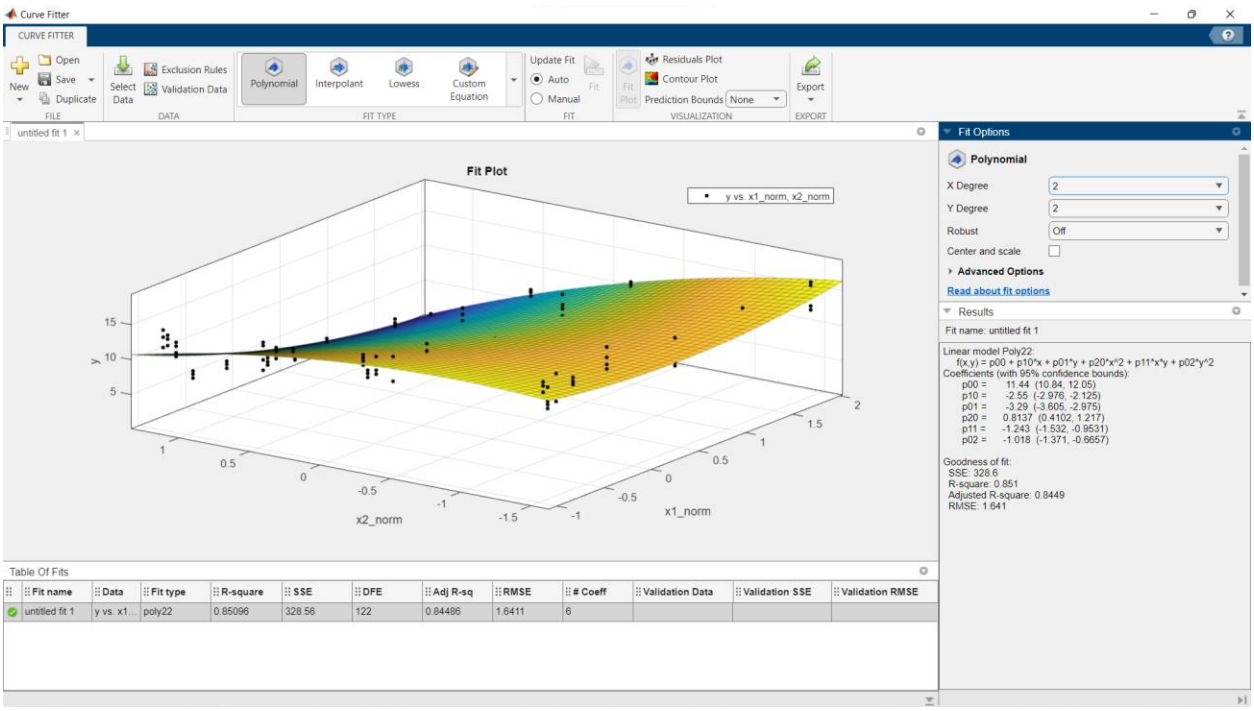

X degree = 4 Y degree=1 R2-Score=0.8325 RMSE = 1.762



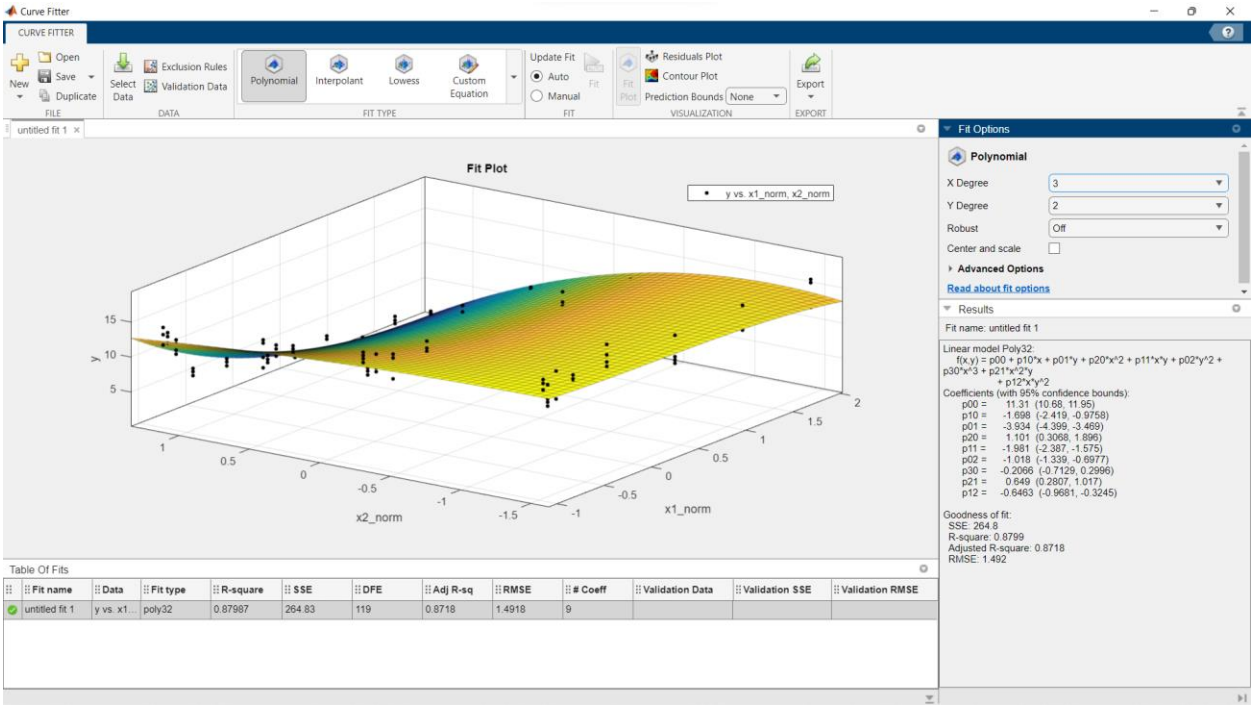

X degree = 5 Y degree=1 R2-Score=0.8483 RMSE = 1.691



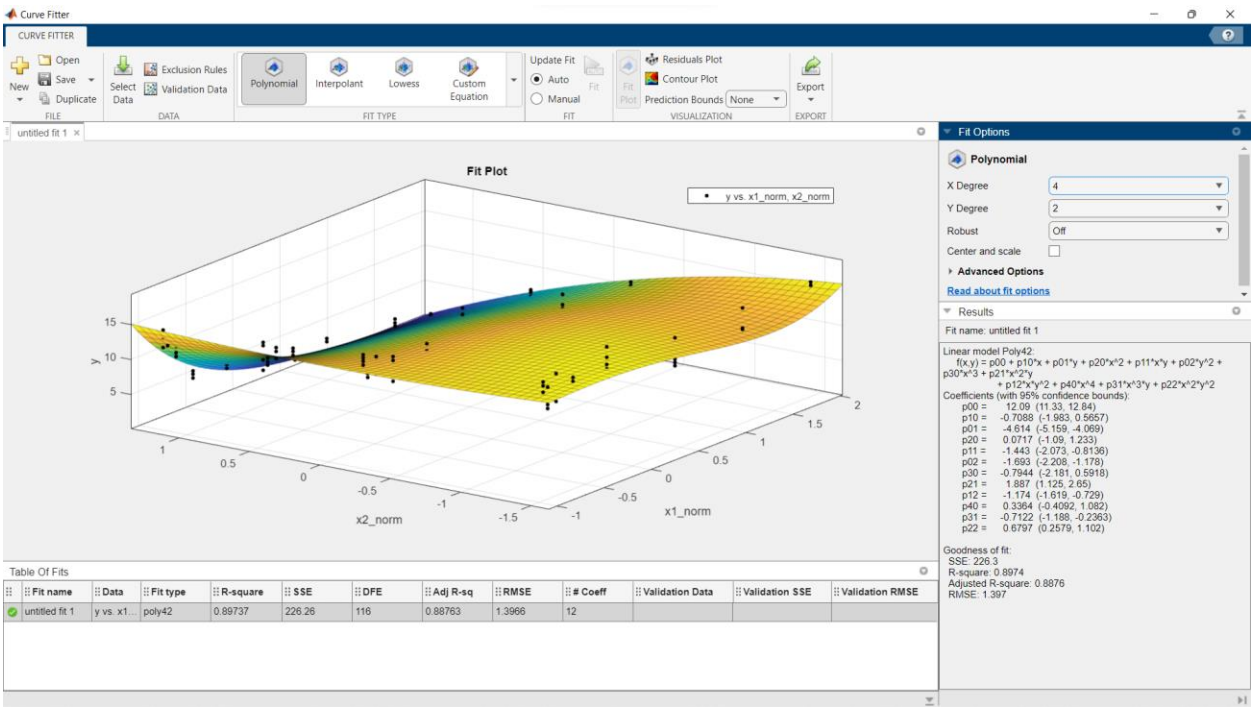

X degree = 1 Y degree=2 R2-Score=0.826 RMSE = 1.738



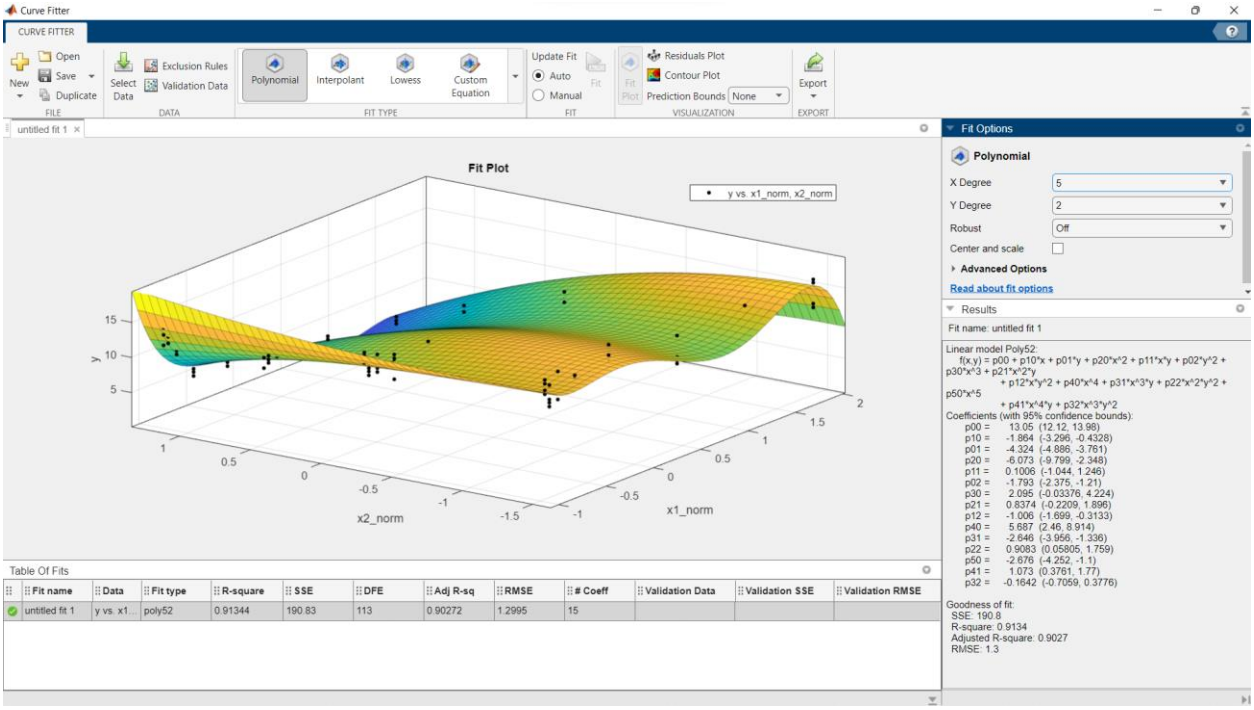

X degree = 2 Y degree=2 R2-Score=0.851 RMSE = 1.641



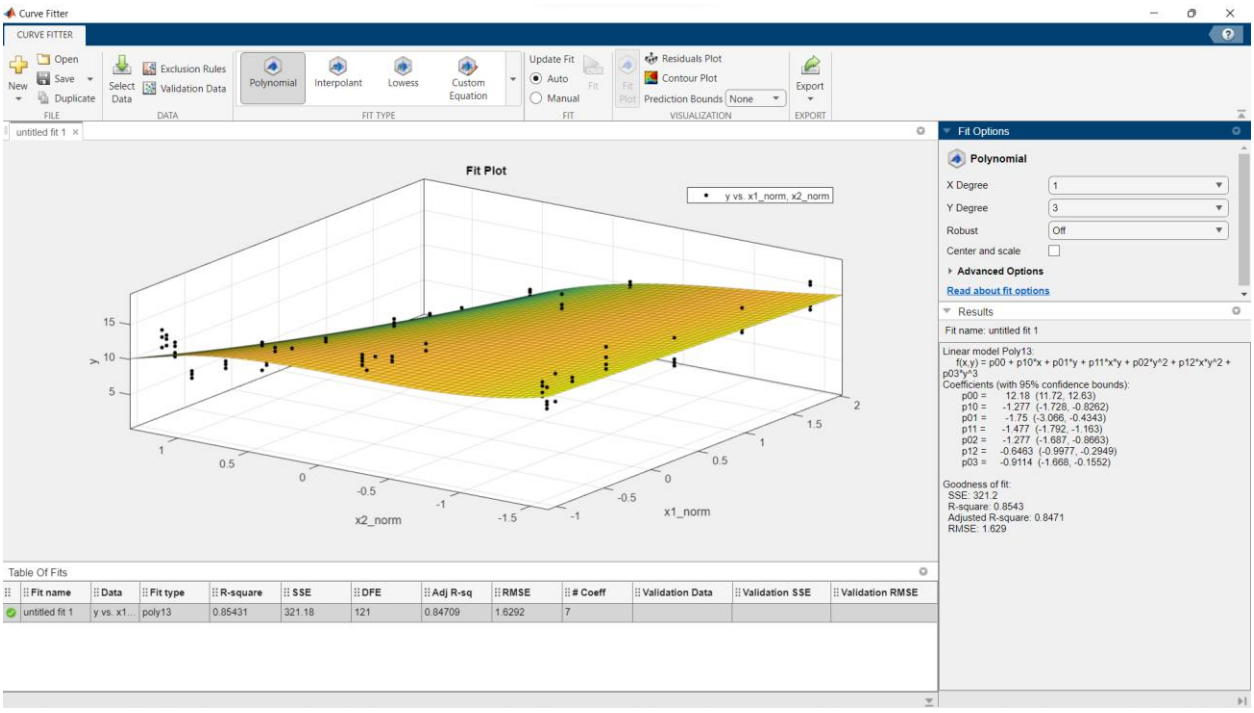

X degree = 3 Y degree=2 R2-Score=0.8799 RMSE = 1.492



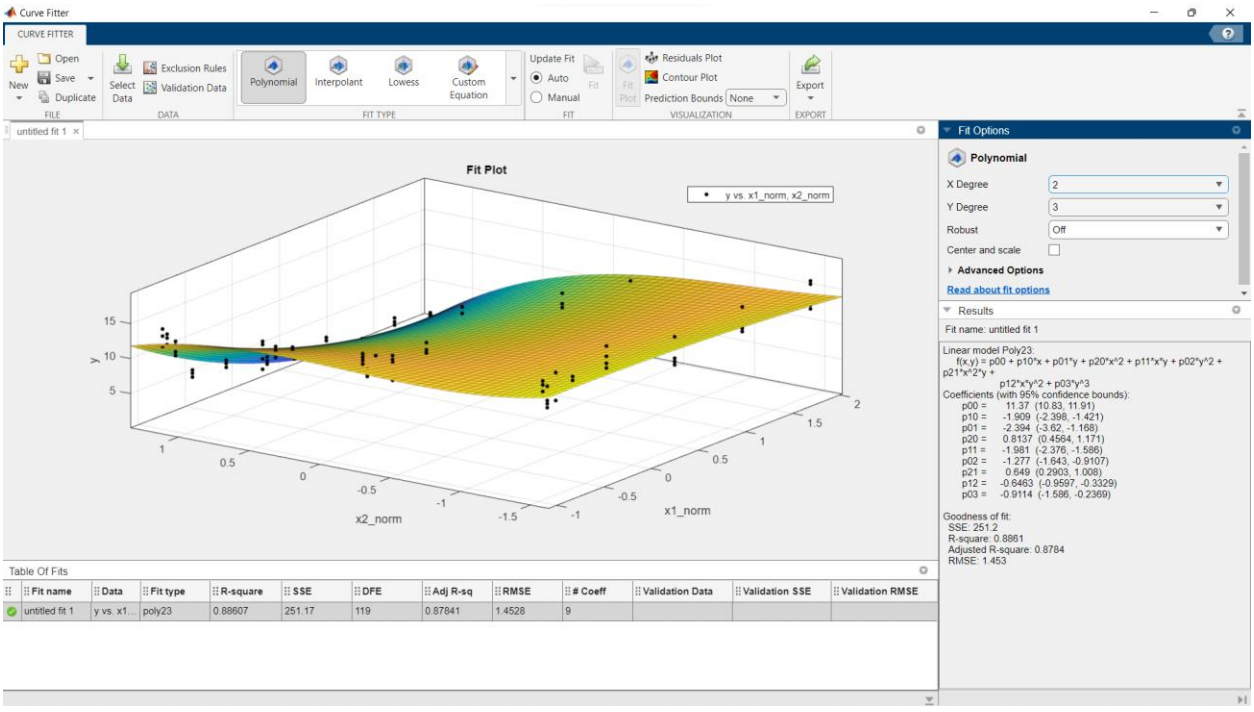

X degree = 4 Y degree=2 R2-Score=0.8974 RMSE = 1.397



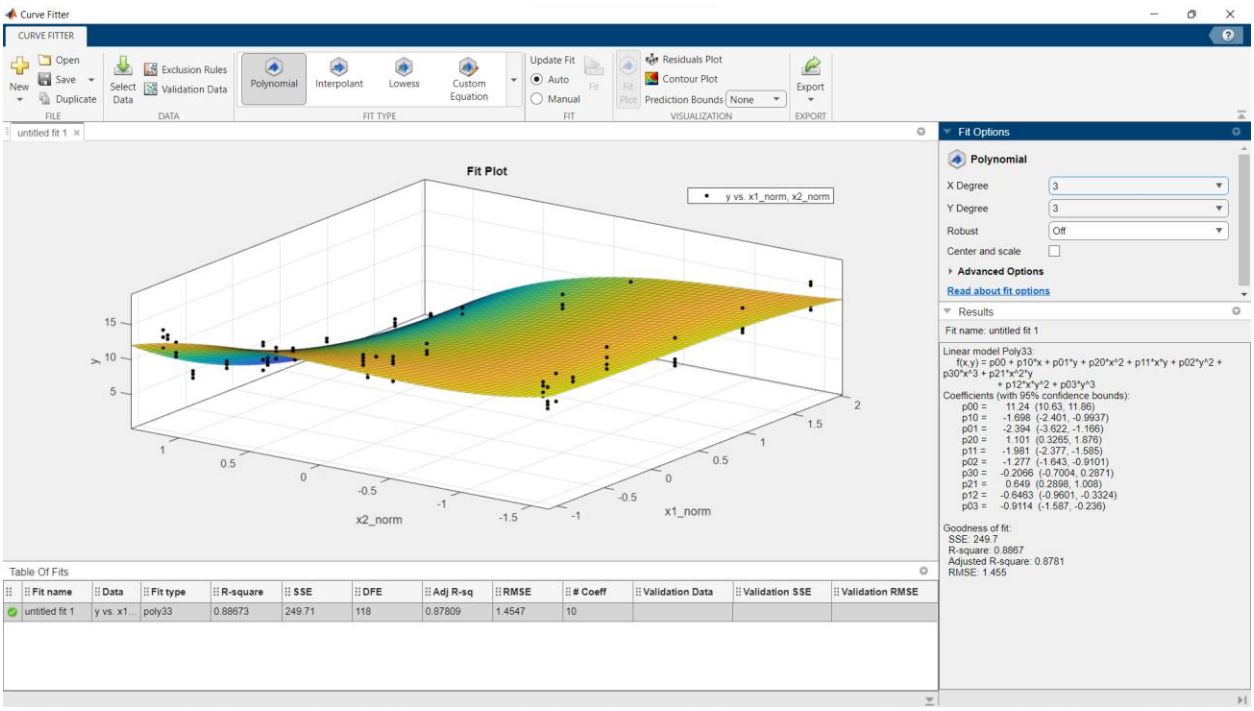

X degree = 5 Y degree=2 R2-Score=0.9134 RMSE = 1.3



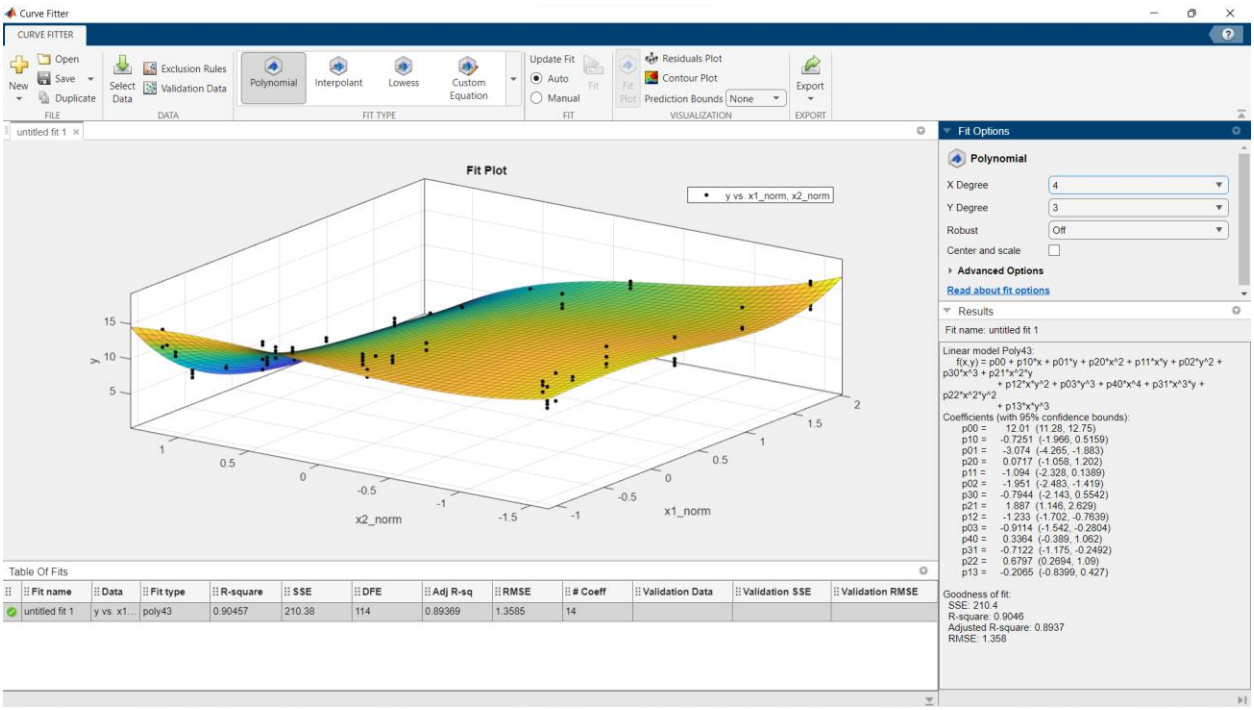

X degree = 1 Y degree=3 R2-Score=0.8543 RMSE = 1.629



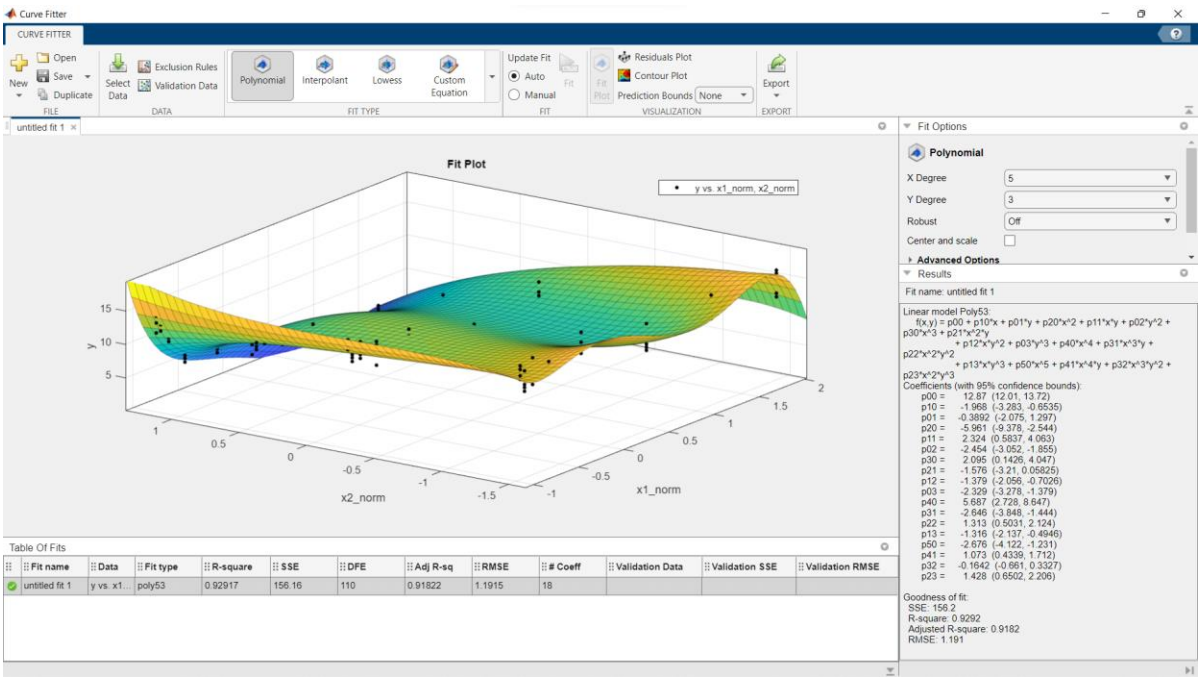

X degree = 2 Y degree=3 R2-Score=0.8861 RMSE = 1.453




X degree = 3 Y degree=3 R2-Score=0.8867 RMSE = 1.455




X degree = 4 Y degree=3 R2-Score=0.9046 RMSE = 1.358




X degree = 5  Y degree=3  R2-Score=0.9292  RMSE = 1.191

**همانگونه که مشاهده می شود با مقایسه ی درجات مختلف برای x و y این نتیجه حاصل می شود که بیشینه R2-Score برای منحنی چند جمله ای با درجات 5 برای x و 3 برای y حاصل می شود. لذا این مدل از لحاظ R2-Score بهترین مدل می باشد.**