Artificial Intelligence

Yalda Amadeh

University of Zanjan

گزارش کار

تمرین سری اول درس یادگیری ماشین – رگرسیون غیرخطی

0) توابع مورد نیاز را در ابتدای برنامه فراخوانی میکنیم.

1) این کد یک معادله خطی با استفاده از پارامترهای تصادفی برای هر یک از چهار مجموعه داده با تعداد داده های مختلف ایجاد می کند. برای هر مجموعه داده، مقدار x را به عنوان ورودی گرفته و بر طبق معادله خطی ارائه شده، با استفاده از پارامترهای تصادفی، مقدار خروجی را محاسبه می کند.

2) این کد 4 مجموعه داده تولید می کند، هر کدام با اندازه دلخواه n_data برای قرار دادن در یک ماتریس دو بعدی. هر داده در مجموعه داده شامل مقادیر x و y است که به ترتیب مقادیر مناسبی از -5 تا 5 را برای x و مقادیر متناظر با عملکرد eq برای y در نظر می گیرند. نتیجه این کد شبیه به یک جدول دو بعدی با دو ستون x و y برای هر مجموعه داده است.

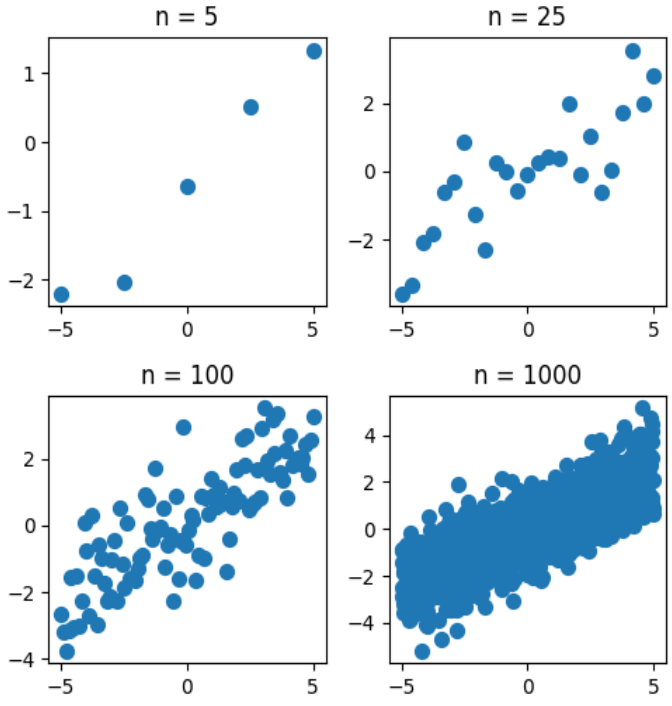
3) این کد یک نمودار 2*2 از نقاط رسم شده با استفاده از روش خوشه بندی در هر کدام از چهار مجموعه داده، به اندازه 5*5 رسم می‌کند. هر خوشه با یک رنگ تفکیک شده است و هر خوشه با نقاطی مشخص شده است که بر اساس دو ویژگی در دو بعد فضا قرار دارند. عنوان (title)هر نمودار هم با توجه به تعداد داده ها (n) آن خوشه تعیین می‌شود.

```python
#0
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```
✓ 2.1s

```python
#1

n_data = [5, 25, 100, 1000]  # تعداد داده ها
# معادله خطی برای هر مجموعه داده، شامل پارامترهای تصادفی
eq = lambda x: 0.5 * x + np.random.randn(len(x))
```
✓ 0.0s

```python
#2
# ایجاد 4 مجموعه داده
data = [np.concatenate((
        np.array([np.linspace(-5, 5, n)]).T,
        eq(np.linspace(-5, 5, n)).reshape(-1, 1)), axis=1)
        for n in n_data]
```
✓ 0.0s

```python
#3
# plot data
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(5,5))
for ax, d, n in zip(axs.flatten(), data, n_data):
    ax.scatter(d[:, 0], d[:, 1], s = 50)
    ax.set_title('n = {}'.format(n))
plt.tight_layout()
plt.show()
```
✓ 0.9s



4) این کد یک لیست از توابع نویز را تعریف می‌کند که هر یک با گرفتن یک عدد n به عنوان ورودی، آرایه‌ای شامل n عنصر تصادفی را با استفاده از توزیع نرمال (Gaussian) و با واریانس‌های مختلف ایجاد می‌کنند. همچنین با تغییر واریانس در هر تابع، می‌توان میزان نویز تولید شده را تنظیم کرد.

5) با تکرار حلقه به ازای هر noise در noises، یک دیتاست با نویز متفاوت تولید می‌شود و به لیستی دیگر به نام datasets اضافه می‌شود. در نهایت، تعداد 4 دیتاست در لیست datasets قرار می‌گیرد. در واضع با این کار به داده های اموزشی مقداری نویز اضافه میشود.

6) در این قسمت از کد برای هر n با مقدار مشخص شده ای noise نمودار های ان را رسم میکنیم.

```
#4
# توابع نویز (تعداد داده به عنوان ورودی)
noises = [lambda n: np.random.randn(n) * 0.2,
          lambda n: np.random.randn(n) * 1.0,
          lambda n: np.random.randn(n) * 5.0]
```

✓ 0.0s

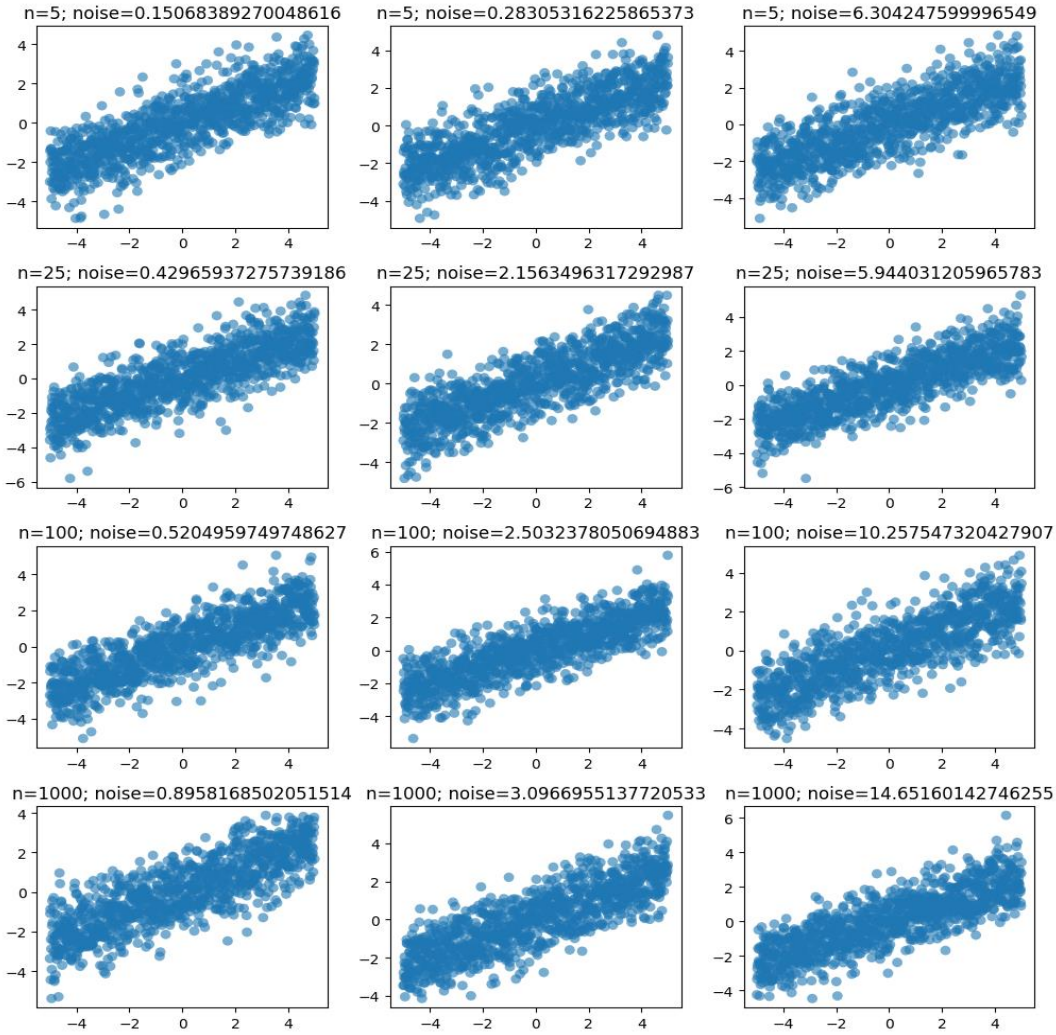
```
#5
# ایجاد 4 مجموعه داده و اضافه کردن نویزهای دلخواه
datasets = []
for n in n_data:
    X = np.linspace(-5,5,n).reshape(-1,1)
    y = eq(np.linspace(-5,5,n)).reshape(-1,1)
    # اضافه کردن نویز به داده ها با حداکثر اندازه نویز 5.0
    dataset = [(X, y+noise(n) if i<len(noises) else y)
               for i, noise in enumerate(noises)]
    datasets.append(dataset)
```

✓ 0.0s

```
#6
# plot data
fig, axs = plt.subplots(nrows=4, ncols=3, figsize=(10,11))
for i, dataset in enumerate(datasets):
    for j, ((X, y), noise) in enumerate(zip(dataset, noises)):
        X = np.linspace(-5,5,n).reshape(-1,1)
        y = eq(np.linspace(-5,5,n)).reshape(-1,1)
        axs[i, j].scatter(X, y, s = 50, alpha=0.6, edgecolors='none')
        axs[i, j].set_title('n={}; noise={}'.format(n_data[i], noise(n_data[i]).max()))
plt.tight_layout()
plt.show()
```

✓ 3.0s

*خروجی مورد انتظار :



7) این کد برای ایجاد سه مدل رگرسیونی با توان چندجمله‌ای 1، 4 و 16 ایجاد شده است.

در نهایت، سه مدل رگرسیونی با درجات مختلف ایجاد شده و در لیستی با نام models قرار داده شده‌اند که می‌توان از آن‌ها برای پیش‌بینی مقادیر بر اساس ورودی‌های داده شده استفاده کرد.

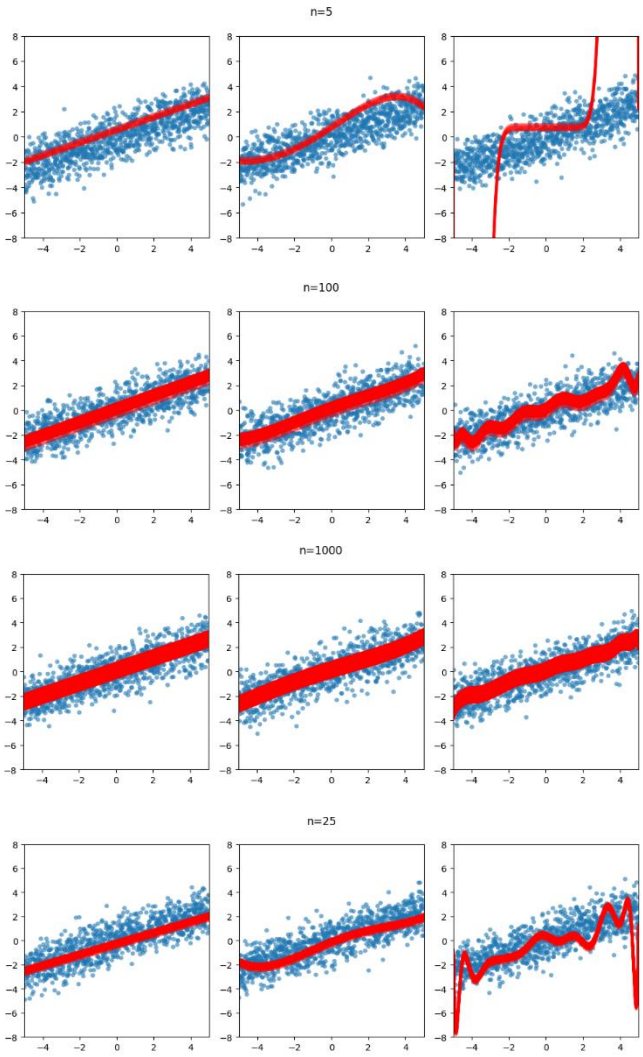
```
#7
# ایجاد 3 مدل رگرسیونی با توان چندجمله‌ای 1، 4 و 16
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression

models = [make_pipeline(PolynomialFeatures(degree=1), LinearRegression()),   # linear regression
          make_pipeline(PolynomialFeatures(degree=4), LinearRegression()),   # polynomial regression degree 4
          make_pipeline(PolynomialFeatures(degree=16), LinearRegression())]  # polynomial regression degree 16
```

✓ 1.5s

8) نمودار تمام خطوط به دست امده با استفاده از هر یک از روش ها برروی همه داده ها:

```
#8
for i, dataset in enumerate(datasets):
    fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(10,4))
    fig.suptitle('n={}'.format(n_data[i]))
    axs = axs.flatten()

    for j, model in enumerate(models):
        X, y = dataset[0]
        model.fit(X, y)

        ax = axs[j]
        X = np.linspace(-5,5,n).reshape(-1,1)
        y = eq(np.linspace(-5,5,n)).reshape(-1,1)
        ax.scatter(X, y, s=25, alpha=0.6, edgecolors='none')

        x_range = np.linspace(-5, 5, 1000).reshape(-1,1)
        y_pred = model.predict(x_range)
        ax.plot(x_range, y_pred, 'r', linewidth=3, alpha=0.6)

        #ax.set_title('Degree = {}, train score = {:.3f}'.format(j*5+1, model.score(X, y)))
        ax.set_ylim([-8, 8])
        ax.set_xlim([-5, 5])
        fig.tight_layout()
```

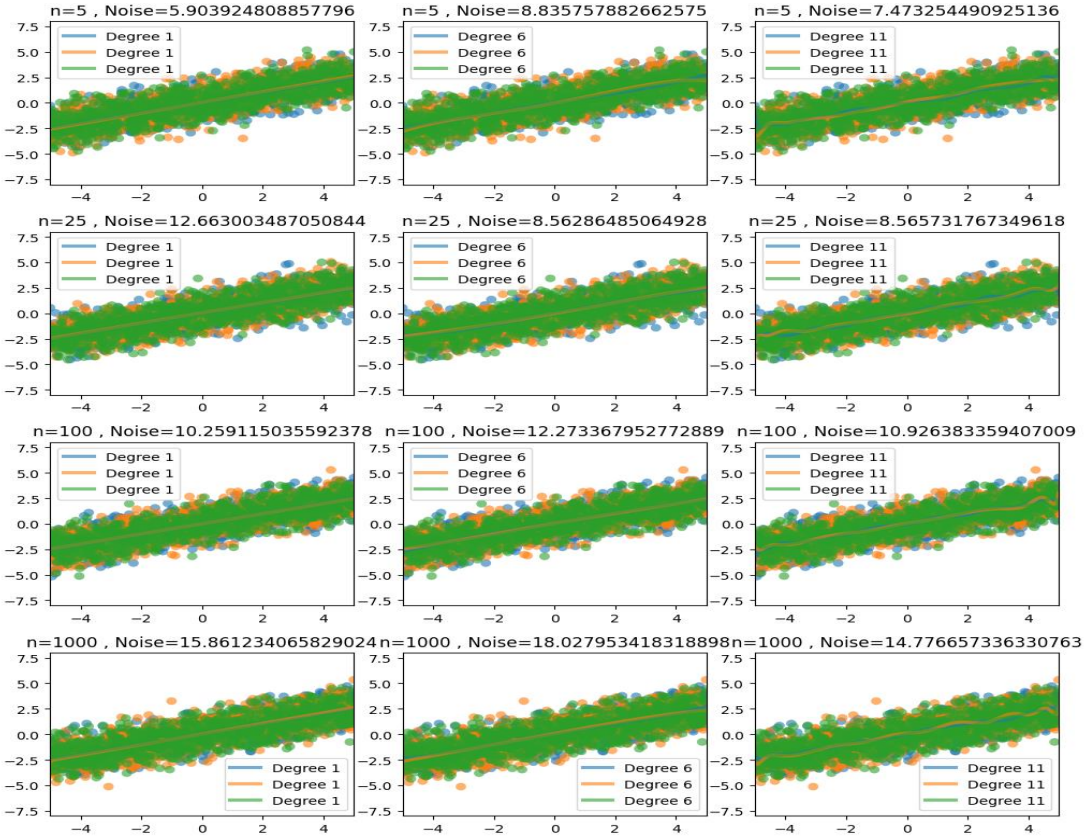
*خروجی مورد انتظار :



9) این کد یک شکل گرافیکی ساختاردهی شده به صورت ۴ سطر و ۳ ستون را برای هریک از دیتاست‌ها از سه مدل مختلف پلات می‌کند. در قسمت اول کد، با استفاده از تابع plt.subplots() یک شکل گرافیکی با ابعاد ۴ در ۳ و با زمینه ای خالی آماده می‌کند.

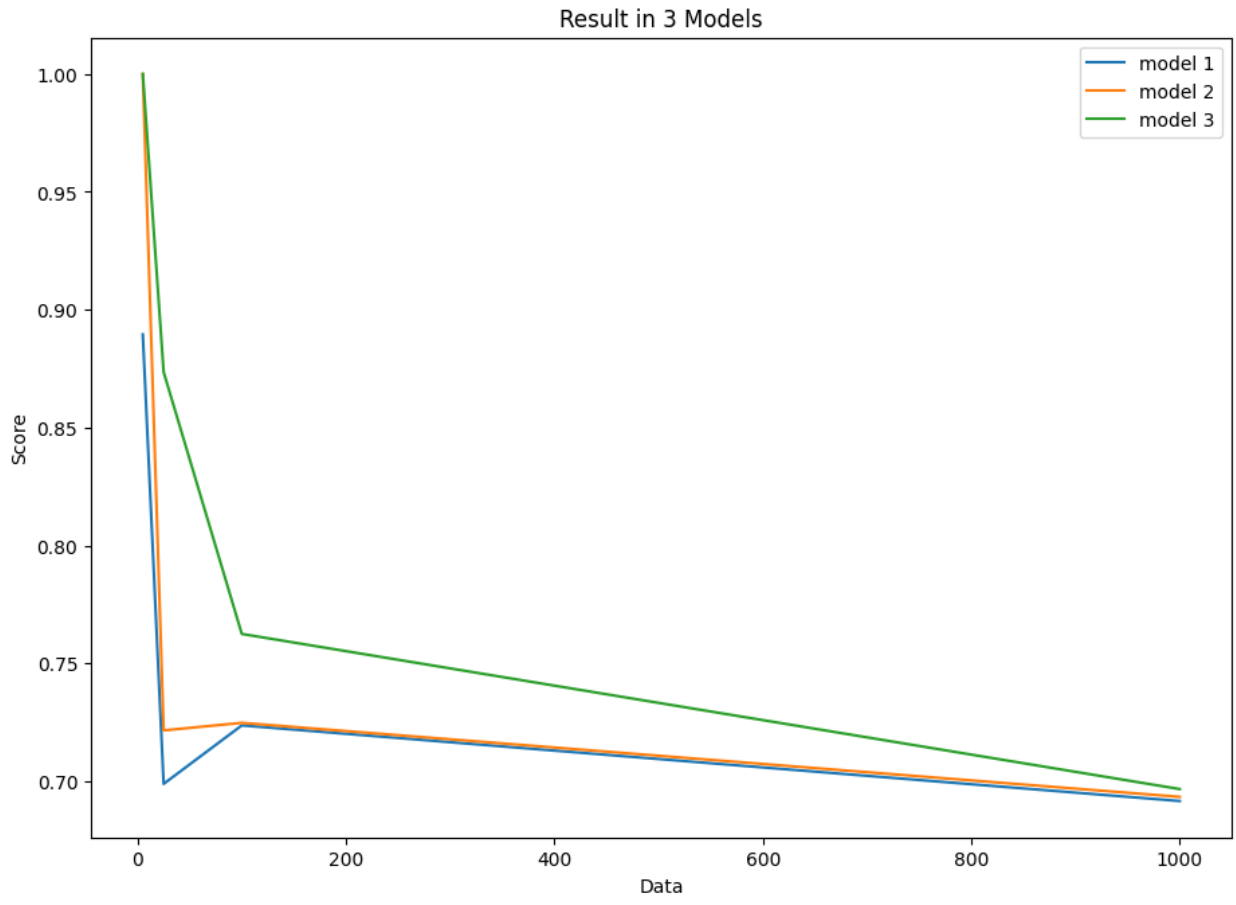
در بخش دوم کد با استفاده از دو حلقه تو در تو به ترتیب دیتاست‌ها و نویزها روی هر یک از مدل‌ها عملیات انجام می‌دهد.در نهایت برای هر کدام از مدل‌ها، از تابع fit() آن استفاده می‌کند تا مقدار پارامترهای مدل را با داده‌های ورودی آموزش داده شده پیدا کند. سپس از مقادیری که پیدا کرده، برای گردشهای مختلف x، خروجی مدل را با استفاده از تابع predict() محاسبه می‌کند. سپس نمودارهای مربوط به هر مدل را با استفاده از تابع‌های plot() و scatter() روی شکل گرافیکی رسم می‌کند.

```
#9
fig, axs = plt.subplots(nrows=4, ncols=3, figsize=(10,11))
for i, dataset in enumerate(datasets):
    for j, ((X, y), noise) in enumerate(zip(dataset, noises)):
        X = np.linspace(-5,5,n).reshape(-1,1)
        y = eq(np.linspace(-5,5,n)).reshape(-1,1)
        for k, model in enumerate(models):
            ax = axs[i, (k % 3)]
            model.fit(X, y)
            x_range = np.linspace(-5, 5, 1000).reshape(-1,1)
            y_pred = model.predict(x_range)
            ax.plot(x_range, y_pred, label=f"Degree {k*5+1}", linewidth=3, alpha=0.6)
            ax.scatter(X, y, s = 50, alpha=0.6, edgecolors='none')
            ax.set_title(f"n={n_data[i]} , Noise={noise(n_data[i]).max()}")
            ax.set_ylim([-8,8])
            ax.set_xlim([-5,5])
            ax.legend()
plt.tight_layout()
plt.show()
```



10) در ادامه کد بالا یک نمودار برای مطالعه دقت نهایی مدل رسم میکنیم که محور افقی حجم دادگان و محور عمودی دقت نهایی بدست امده از مدل است . در این نمودار 3 خط با 3 رنگ مختلف رسم میکنیم که نشان دهنده هر یک از مدل های تولید شده است.

```
#10
# حاشیه نمودار
plt.figure(figsize=(10,7))
plt.subplots_adjust(left=0.1, right=0.95, bottom=0.1, top=0.95, hspace=0.3, wspace=0.3)

# رسم نمودار
for i, model in enumerate(models):
    r_squared_list = []
    for j in range(len(datasets)):
        X_train, y_train = datasets[j][i]
        model.fit(X_train, y_train)
        X_test, y_test = datasets[j][i]
        y_pred = model.predict(X_test)
        r_squared = model.score(X_test, y_test)
        r_squared_list.append(r_squared)

    plt.plot(n_data, r_squared_list, label=f'model {i+1}')

plt.xlabel('Data')
plt.ylabel('Score')
plt.title("Result in 3 Models")
plt.legend()
plt.show()
```



پاسخ به سوالات انتهایی :

1) برای دادگان خطی ، ایا استفاده از رگرسیون غیرخطی مناسب است و دقت کافی دارد ؟ در چه شرایطی این روش جواب مناسب نخواهد بود ؟

- برای داده های خطی، استفاده از روش‌های خطی مناسب تر است. اما در برخی موارد، استفاده از روش های غیرخطی مانند رگرسیون غیرخطی نیز می تواند جواب مناسبی داشته باشد. استفاده از روش های رگرسیون غیرخطی مانند رگرسیون چند جمله ای در بعضی موارد می تواند حتی به کاهش دقت منجر شود. در شرایطی که داده ها اصلاً خطی نیستند و رابطه بین ورودی و خروجی پیچیده است، ممکن است رگرسیون غیرخطی بهترین گزینه باشد.

اگر داده ها حاوی رفتارهای غیرخطی باشند، مانند رفتار سیگموئید یا تابع لجستیک، استفاده از رگرسیون غیرخطی بهترین گزینه است. همچنین در مواردی که داده ها حاوی انواع عدم تقارن، انحراف، یا انحراف کوچک از خطی بودن باشند، می توانید از رگرسیون غیرخطی استفاده کنید.

با این وجود، در برخی شرایط، روش رگرسیون غیرخطی مناسب نخواهد بود، به عنوان مثال:

- زمانی که داده ها بسیار کم هستند و خطاهای زیادی در تخمین پارامترهای مدل رخ می دهد. در این شرایط، استفاده از رگرسیون خطی بهتر است.

- زمانی که داده ها حاوی انواع نویز و اشتباهات زیادی هستند.

بنابراین، استفاده از رگرسیون غیرخطی بستگی به خصوصیات داده ها و شرایط مسئله دارد و باید با توجه به شرایط مسئله انجام شود.

2) اگر هیچ دانشی در مورد دادگان خود نداشته باشیم ، چطور می توان توان چند جمله ای لازم را بیابیم؟ (فرض کنید دادگان ما دارای تعداد زیادی ویژگی هستند که رسم انها را غیر ممکن می سازد)

برای تعیین توان چند جمله ای لازم، می توان از روش های مختلف مانند رگرسیون چند جمله ای با درجه مختلف استفاده کرد و سپس با مقایسه دقت هر مدل روی داده های آزمایشی، تصمیم گیری کرد. همچنین می توان از روش های مشابه مانند الگوریتم های یادگیری ماشینی برای تعیین توان چند جمله ای استفاده کرد.

برای پیدا کردن توان چند جمله‌ای لازم برای دادگانی که دارای ویژگی‌های زیادی هستند، می‌توان از روش PCA یا Principal Component Analysis استفاده کرد. با این روش، ابتدا تمام ویژگی‌های دادگان را به چند ویژگی کمرنگ‌تر تبدیل می‌کنیم که هر کدام از آن‌ها ترکیبی از ویژگی‌های اصلی دادگان هستند. سپس با تحلیل تغییرات این ویژگی‌های کمرنگ‌تر، می‌توان توان چند جمله‌ای لازم برای توصیف دادگان اصلی را پیدا کرد.

3) ایا کاهش خطای مدل در زمان اموزش همواره مطلوب است ، ایا امکان دارد مدلی که دقت اموزشی کمتری دارد ، در زمان استفاده بدتر عمل کند؟ چرا؟

کاهش خطای مدل در زمان آموزش معمولاً مطلوب است، زیرا این نشان دهنده بهبود کارایی مدل در پیش‌بینی داده‌های جدید است. مثال، در شرایطی که مدل بسیار پیچیده است و تعداد پارامترهای آن بسیار زیاد است، افزایش دقت در مرحله آموزش می تواند منجر به Overfitting شود و در نتیجه دقت در پیش بینی داده های جدید را کاهش دهد. علاوه بر این، در بعضی موارد محدودیت های زمانی و منابع محاسباتی ممکن است حداقل خطای آموزشی را به جای حداکثر کردن دقت پیش بینی داده های جدید قرار دهد. برای این دلیل، لازم است که در هر مورد به طور جدی به مقدار خطای آموزشی و دقت پیش بینی داده های جدید توجه شود و مدلی که خطای آموزشی کمتری دارد، بهتر استفاده شود، به شرطی که دقت پیش بینی در داده های جدید هم خوب باشد. در واقع، هدف اصلی ما باید یافتن یک مدل با عملکرد خوب برای داده‌های جدید باشد نه فقط ارائه یک مدل با دقت بالا برای داده‌های آموزشی.

پایان