به نام خدا

1928
K. N. Toosi University of Technology

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

پاسخنامه میان‌ترم

محمدرضا جنیدی جعفری

۹۹۲۵۲۵۳

لینک کولب

استاد : آقای دکتر مهدی علیاری

پاییز ۱۴۰۳

فهرست مطالب

عنوان شماره صفحه

بخش ۱: سوالات هماهنگ شده

سوال اول

الف)

قبل از نوشتن معادلات، لازم است هر کدام از ویژگی ها را معرفی کنیم:

TP:

موارد به درستی طبقه‌بندی‌شده به عنوان کلاس هدف

TN:

مواردی که متعلق به کلاس هدف نیستند و به درستی طبقه‌بندی شده‌اند

FP:

مواردی که به اشتباه به عنوان کلاس هدف طبقه‌بندی شده‌اند

FN:

مواردی که متعلق به کلاس هدف هستند اما به اشتباه طبقه‌بندی شده‌اند

حالا در زیر، معادلات Sensitivity و Specificity را می نویسیم:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

ب)

	C_1'	C_2'	C_3'	C_4'
C_1	45	3	2	1
C_2	3	32	2	6
C_3	2	2	16	10
C_4	0	2	0	20

نتایج را در جدول زیر آورده ام:

Class	TP	FP	FN	TN
C1	45	5	6	90
C2	32	7	11	96
C3	16	4	14	112
C4	20	17	2	107

Class	Sensitivity	Specificity
C1	0.882352941	0.947368421
C2	0.744186047	0.932038835
C3	0.533333333	0.965517241
C4	0.909090909	0.862903226

سوال دوم

قبل از حل سوال، بهتر است مفهوم Back Propagation را بررسی کوتاهی کنیم:

Backpropagation یا پس‌انتشار خطا یکی از الگوریتم‌های اساسی در یادگیری شبکه‌های عصبی مصنوعی است که برای تنظیم وزن‌ها و بایاس‌ها در شبکه استفاده می‌شود. این روش باعث می‌شود شبکه بتواند به‌صورت خودکار و بهینه خطای پیش‌بینی را کاهش دهد.

شبکه ای که در صورت سوال معرفی شده است به صورت زیر است:

$$2 \times 3 \times 2 \times 1$$

به این معنا که لایه ورودی دارای ۲ نورون است، لایه پنهانی اول دارای ۳ نورون، لایه پنهانی دوم دارای ۲ نورون و لایه نهایی(خروجی) ۱ نورون دارد.

برای حل سوال نیاز به چند فرض داریم که در زیر آورده شده است:

فرض ۱- وزن‌ها (w) و بایاس‌ها (b) برای هر نورون با مقادیر فرضی مقداردهی می‌شوند.

فرض ۲- از تابع فعالسازی سیگموید استفاده میکنیم:

$$\sigma(z) = \frac{1}{1+ e^{-z}}$$

فرض ۳- نرخ یادگیری اتا مشخص شده است.

فرض ۴- از تابع میانگین خطای مربع (MSE) استفاده می کنیم:

$$Error = \frac{1}{2}(y_{real} - y_{pred})^2$$

مراحل محاسبه، به مراحل Forward Pass، محاسبه خطا و Backward Pass تقسیم میشود:

Forward Pass:

- سیگنال ورودی از لایه ورودی به لایه‌های پنهان و سپس به لایه خروجی منتقل می‌شود.
- خروجی هر نمونه محاسبه می شود:

$$z = b + \Sigma(x.w)$$

$$\sigma(z) = 0$$

Loss Calculation:

$$Error = \frac{1}{2}(y_{real} - y_{pred})^2$$

Backward Pass:

- محاسبه خطای نورون خروجی:

$$\sigma'(z).(y_{real} - y_{pred}) = \delta_{out}$$
$$\sigma'(z) = (\sigma(z) - 1).\sigma(z)$$

- به روز رسانی وزن های مرتبط با لایه خروجی:

$$w_{new} = o_{prev\,n}.\delta_{out}.\eta - w_{old}$$

برای هر نورون در لایه پنهان:

- خطای نورون بر اساس خطاهای نورون های بعدی محاسبه می شود:

$$\delta_{hidden} = \sigma'(z).(w_{new\,L}.\Sigma\delta_{out})$$

- به روز رسانی وزن ها و بایاس ها:

$$w_{new} = o_{prev\,n}.\delta_{out}.\eta - w_{old}$$

حل یک مثال:

Input Layer(2 n) : x_1, x_2

Hidden Layer 1(3 n): $h_{1,1}, h_{1,2}, h_{1,3}$

Hidden Layer 2(2 n): $h_{2,1}, h_{2,2}$

Output Layer(1n): o

فرض:

$x_1 = 0.5 , x_2 = 0.9$

Weight (random):

$w_{11} = 0.1 , w_{12} = 0.2 , w_{13} = 0.3 , w_{21} = 0.4 , w_{22} = 0.5 , w_{23} = 0.6$

$w_{h1} = 0.1 , w_{h2} = 0.2 , w_{h3} = 0.3 , w_{h4} = 0.4 , w_{h5} = 0.5 , w_{h6} = 0.6$

$w_{o1} = 0.3 , w_{o2} = 0.7$

Biases:

Hidden Layer 1: $b_{h1} = 0.1 , b_{h2} = 0.1 , b_{h3} = 0.1$

$Hidden\ Layer\ 2$: $b_{h4} = 0.2 , b_{h5} = 0.5$

Output Layer: $b_o = 0.1$

Learning Rate: $\eta = 0.01$

$y_{true} = 1$

Step 1: Forward Pass

$z_{h1,1} = 0.51$

$o_{h1,1} = 0.625$

$z_{h1,2} = 0.64 , o_{h1,2} = 0.654$

$z_{h1,3} = 0.77 , o_{h1,3} = 0.683$

$z_{h2,1} = 1.634 , o_{h2,1} = 0.836$

$z_{h2,2} = 1.003 , o_{h2,2} = 0732$

$z_o = 0.841$, o = 0.698

Step 2: Compute the Error

E = 0.045

Step 3: Backward Pass

$\sigma'(z_o) = 0.211$

$\delta_o = -0.0637$

$\sigma'(z_{h2,1}) = 0.137$

$\delta_o = -0.00262$

$\delta_{h2,2} = -0.00651$

$\sigma'(z_{h1,1}) = 0.234$

$\delta_{h1,1} = -0.00066$

$w_{11,new} = 0.1000033$

بخش۲: سوالات هماهنگ نشده

سوال دوم

الف)

شبکه هافیلد (Hopfield Network) از نوع شبکه‌های عصبی بازگشتی است که برای ذخیره و بازیابی الگوها طراحی شده است. اگر نشانه‌های گره‌ها (یعنی علامت مثبت و منفی) معکوس شوند، همچنان همان الگوی اصلی باقی می‌ماند.

دلایلی که می شود اشاره کرد:

۱. تعریف تابع انرژی
۲. تقارن وزنی
۳. خاصیت پایداری دینامیک سیستم

تابع انرژی در شبکه هاپفیلد به صورت زیر است:

$$E = -\frac{1}{2}\sum_{i}\sum_{j} w_{ij}s_i s_j + \sum_i \theta_i s_i$$

با ساده سازی تابع بالا، درنهایت به رابطه ی $E = E'$ می رسیم که نشان می‌دهد که انرژی شبکه بدون تغییر باقی می‌ماند و سیستم همچنان در همان حالت پایدار (یا مینیمم انرژی) قرار دارد.

از طرفی در شبکه هاپفیلد داریم:

$$W_{ij} = W_{ji}$$

این تقارن تضمین می‌کند که انرژی سیستم تنها به حالت نسبی گره‌ها بستگی دارد، نه به نشانه‌های خاص گره‌ها.

وقتی تمام گره‌ها معکوس شوند، تأثیری روی محاسبات انرژی یا پایداری الگوها ندارد، زیرا روابط متقابل بین گره‌ها (که توسط وزن‌ها تعریف می‌شوند) تغییر نمی‌کند.

ب)

هاپفیلد

مزایا:

۱. **حافظه تداعی‌گر *(Associative Memory)*:**

- شبکه هافیلد می‌تواند الگوها را ذخیره و با استفاده از نشانه‌های ناقص یا نویزی، آن‌ها را بازیابی کند.
- مثال: اگر یک تصویر ناقص از حرف *"A"* به شبکه داده شود، می‌تواند تصویر کامل *"A"* را بازسازی کند.

۲. **پایداری انرژی:**

- شبکه در حالت پایدار قرار می‌گیرد (حداقل انرژی). این تضمین می‌کند که الگوهای ذخیره‌شده به عنوان حالات پایدار باقی می‌مانند.

۳. **سادگی در پیاده‌سازی:**

- قوانین یادگیری در شبکه هافیلد ساده و مستقیم هستند (مانند قانون هب).

مشکلات:

۱. **ظرفیت ذخیره‌سازی محدود:**

- شبکه هافیلد تنها می‌تواند تقریباً $0.15 \times N$ *0.15×N0.15 \times N0.15×N* الگو ذخیره کند *(NNN* تعداد گره‌ها). اگر تعداد الگوها زیاد شود، بازیابی صحیح مختل می‌شود.

۲. **نویز و هم‌پوشانی الگوها:**

- در صورت ذخیره الگوهای بسیار مشابه، شبکه ممکن است به الگوهای جدید یا ترکیبی *(spurious states)* برسد.

۳. **کندی در یادگیری:**

- آموزش شبکه برای تعداد زیادی الگو و گره زمان‌بر است.

۴. **محدودیت در انواع مسائل:**

- شبکه هافیلد بیشتر برای مسائل حافظه تداعی‌گر مناسب است و نمی‌تواند برای همه مسائل یادگیری ماشین استفاده شود.

مثال:

فرض کنید می‌خواهید سه الگو(ABC) را در شبکه‌ای با ۱۰ گره ذخیره کنید. شبکه می‌تواند این الگوها را ذخیره کند، اما اگر الگوی جدیدی بسیار مشابه با Aباشد، ممکن است شبکه به اشتباه Aرا به جای الگوی جدید بازگرداند.

رگرسیون لجستیک *(Logistic Regression)*

مزایا:

۱. **سادگی و سرعت:**
   - رگرسیون لجستیک بسیار ساده است و الگوریتم آن سریع اجرا می‌شود.
۲. **مناسب برای طبقه‌بندی دودویی:**
   - رگرسیون لجستیک برای مسائلی با دو کلاس بسیار مناسب است.
۳. **تفسیرپذیری:**
   - ضرایب مدل قابل تفسیر هستند و تأثیر هر ویژگی را بر نتیجه پیش‌بینی نشان می‌دهند.
۴. **مقاومت در برابر *Overfitting*:**
   - با استفاده از تکنیک‌های تنظیم *(Regularization)* مانند $L1L1L1$یا $L2L2L2$، از بیش‌برازش جلوگیری می‌شود.

مشکلات:

۱. **فرض خطی بودن:**
   - رگرسیون لجستیک فرض می‌کند که بین ویژگی‌ها و خروجی رابطه خطی وجود دارد. این در بسیاری از مسائل پیچیده صادق نیست.
۲. **محدودیت در طبقه‌بندی چندگانه:**

- رگرسیون لجستیک ذاتاً برای طبقه‌بندی دودویی طراحی شده و برای طبقه‌بندی چندکلاسه نیاز به گسترش دارد) مانند.*Softmax)*

۳. **حساسیت به مقادیر پرت*:(Outliers)***

- وجود داده‌های پرت می‌تواند ضرایب را به شدت تحت تأثیر قرار دهد.

۴. **نیاز به مقیاس‌بندی داده‌ها:**

- اگر داده‌ها مقیاس‌بندی نشده باشند، سرعت همگرایی و عملکرد مدل ممکن است مختل شود.

مثال:

فرض کنید می‌خواهید پیش‌بینی کنید که آیا دانش‌آموزی در آزمون قبول می‌شود یا خیر (طبقه‌بندی دودویی). ویژگی‌ها شامل تعداد ساعت‌های مطالعه و نمره‌های پیشین هستند. رگرسیون لجستیک می‌تواند احتمال قبولی دانش‌آموز را پیش‌بینی کند (خروجی بین *0* و *1*)

سوال سوم

با استفاده از کدر زیر معادلات خطوط را استخراج می کنیم:

```
import numpy as np

# تعریف نقاط
points = np.array([[2, 2], [2, 6], [4, 6], [6, 8], [6, 2], [2,
2]])  # اصلاح خطا

# محاسبه معادلات خطوط
lines = []
for i in range(len(points) - 1):
    x1, y1 = points[i]
    x2, y2 = points[i + 1]
    A = y1 - y2
    B = x2 - x1
    C = x1 * y2 - x2 * y1
    lines.append((A, B, C))

# نمایش معادلات خطوط
for i, (A, B, C) in enumerate(lines):
    print(f"Line {i + 1}: {A}x + {B}y + {C} = 0")
```

خروجی:

Line 1: -4x + 0y + 8 = 0

Line 2: 0x + 2y + -12 = 0

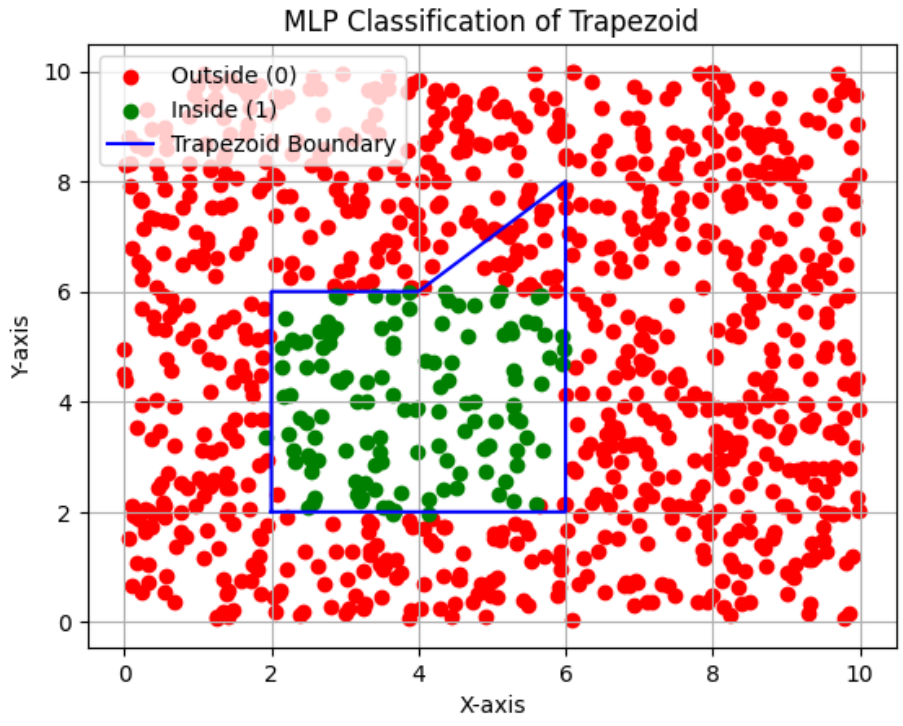
Line 3: -2x + 2y + -4 = 0

Line 4: 6x + 0y + -36 = 0

Line 5: 0x + -4y + 8 = 0

این کلاس یک شبکه عصبی با سه لایه تعریف می‌کند:

- **لایه اول :(hidden1)** شامل ۵ نورون است که وزن‌ها و بایاس‌ها به صورت دستی مقداردهی می‌شوند تا با خطوط مرزی ذوزنقه هماهنگ شوند.
- **لایه دوم :(hidden2)** شامل ۲ نورون است و وزن‌ها به صورت تصادفی مقداردهی می‌شوند.
- **لایه خروجی :(output)** شامل ۱ نورون است و خروجی احتمال نقطه‌ای در داخل یا خارج ذوزنقه را پیش‌بینی می‌کند.



متاسفانه دو ۲۰ دقیقه زدم و وقت ندارم دیگر.

سوال چهارم

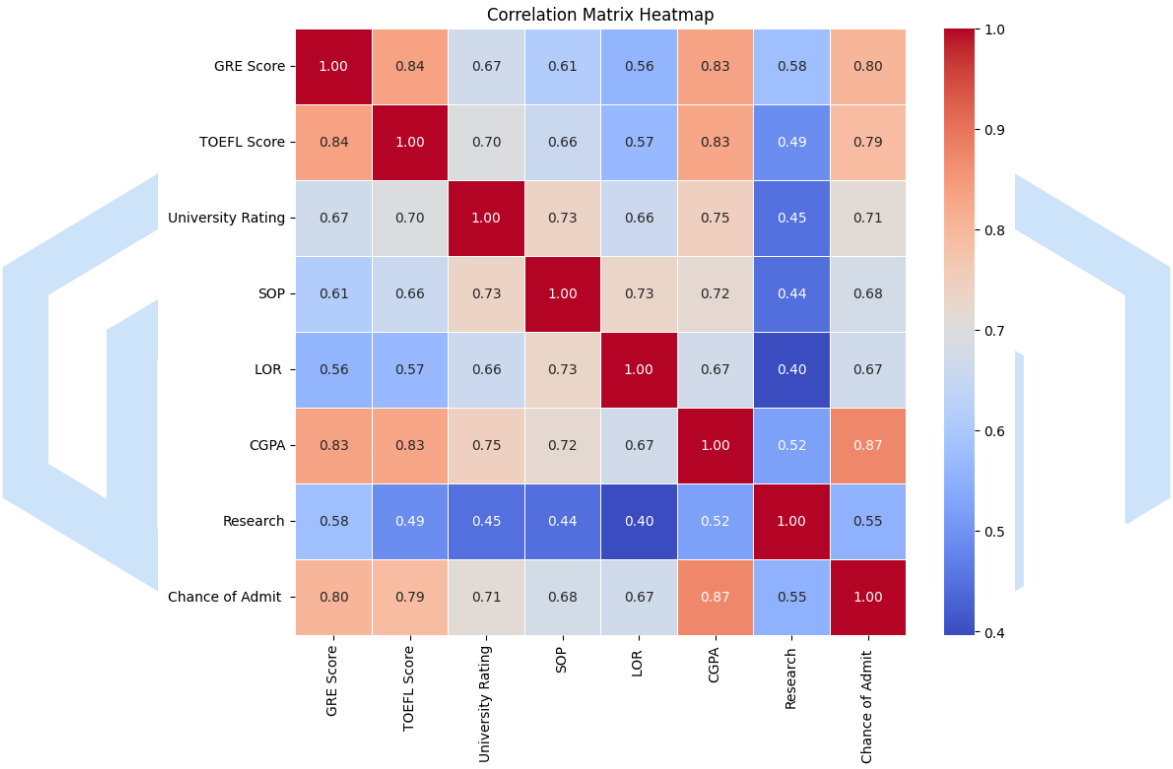
الف)

ابتدا با استفاده از دستور gdown و کتابخانه pandas داده را میخوانیم:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

داده دارای ۴۰۰ نمونه، ۷ ویژگی و یک داده هدف است که داده هدف، یک احتمال بین ۰ و یک است.

ماتریس زیر، همبستگی میان ویژگی و شانس پذیرش را نمایش می دهد:



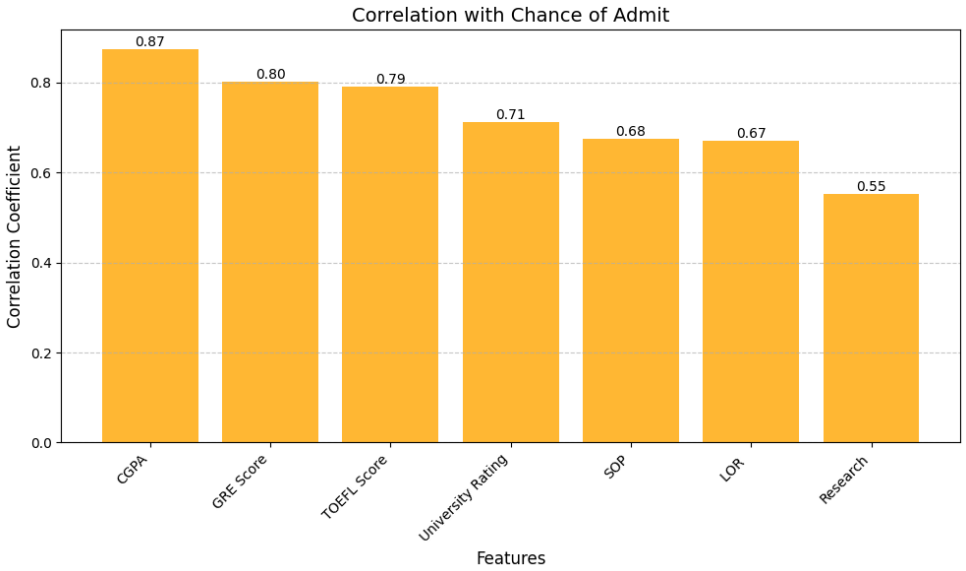
حالا میزان همبستگی میان هر ویژگی را با شانس پذیرش با استفاده از کد زیر نشان می دهیم:

```
correlation_with_admit = correlation_matrix["Chance of Admit
"].drop("Chance of Admit ").sort_values(ascending=False)
```

و کد زیر آنرا به صورت نزولی نمایش می دهد:

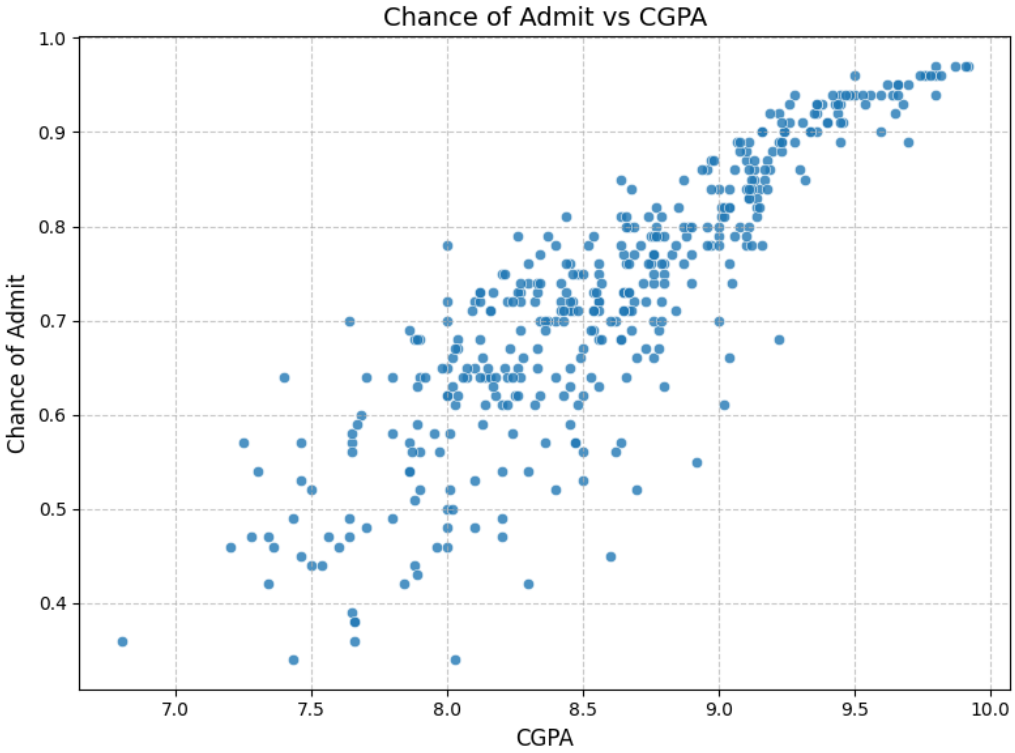
```
highest_correlation_feature = correlation_with_admit.idxmax()
```

نمودار میله ای آن به صورت زیر است:



واضح است GPA (معدل) و نمره GRE بیشترین تاثیر را در شانس پذیرش در دانشگاه UCLA دارند و تجربه تحقیقاتی و کیفیت توصیه نامه ها کمترین تاثیر را دارند.

حالا، توزیع ویژگی CGPA و شانس پذیرش را رسم می کنیم:



میتوان تحلیل زیر را از این نمودار داشت:

۱. **رابطه مثبت قوی:**

- نقاط پراکندگی در نمودار نشان می‌دهند که هرچه مقدار CGPA افزایش پیدا کند، شانس پذیرش نیز به طور قابل توجهی افزایش می‌یابد.
- این ادعا نشان‌دهنده یک رابطه مثبت قوی بین این دو متغیر است که با ضریب هم‌بستگی بالای ۰.۸۷ تأیید می‌شود.

۲. **الگوی خطی:**

- نقاط به طور تقریبی یک الگوی خطی را دنبال می‌کنند. این نشان می‌دهد که احتمال پذیرش به صورت خطی با CGPA تغییر می‌کند.

۳. **پراکنده بودن در مقادیر پایین‌تر:**

- در مقادیر پایین CGPA (۷ تا ۸)، شانس پذیرش گستردگی بیشتری دارد. این ممکن است نشان‌دهنده عوامل دیگری باشد که در این محدوده تأثیر بیشتری دارند، مثلاً تجربه تحقیقاتی یا توصیه‌نامه‌ها و....

۴. **پیش‌بینی‌پذیری بالا در مقادیر بالا:**

- در مقادیر بالای CGPA (بیش از ۹)، شانس پذیرش بسیار متمرکز و نزدیک به ۱ است. این نشان می‌دهد که دانشجویانی با CGPA بالا به احتمال بسیار زیاد پذیرفته خواهند شد.

ب)

قبل از تقسیم داده، کمی داده هدف را تحلیل می کنیم. در حین تقسیم بندی نمی شود از satisfying استفاده کرد زیرا تعداد نمونه هایی خیلی کم وجود دارد.

تحلیل نیاز به نرمال‌سازی یا استانداردسازی:

- **نرمال‌سازی:(Normalization)**
  - اگر مقادیر متغیرها در بازه‌های مختلفی قرار داشته باشند (مثلاً GRE از ۰ تا ۳۴۰ و Researchبین ۰ و ۱)، بهتر است از نرمال‌سازی استفاده کنید تا همه ویژگی‌ها در بازه‌ای مشابه (مثلاً ۰ تا ۱) باشند.
- **استانداردسازی:(Standardization)**

- اگر مقیاس‌بندی داده‌ها بر اساس میانگین و انحراف معیار مهم باشد، استانداردسازی ترجیح داده می‌شود. این روش مقادیر را به صورتی تغییر می‌دهد که میانگین صفر و انحراف معیار یک داشته باشند.

نتیجه‌گیری:

با توجه به داده‌های موجود (مانند GRE و GPA که مقیاس‌های مختلفی دارند)، استفاده از **نرمال‌سازی** مناسب‌تر به نظر می‌رسد.

برای تقسیم داده‌ها به مجموعه‌های آموزش و آزمون و انجام نرمال‌سازی، باید نرمال‌سازی را **به‌طور جداگانه روی مجموعه‌های آموزش و آزمون** اعمال کنیم. دلیل اصلی این کار این است که:

۱. **مدل نباید اطلاعاتی از داده‌های آزمون داشته باشد** :اگر داده‌های آزمون در فرآیند نرمال‌سازی استفاده شوند (به عنوان مثال برای محاسبه حداقل و حداکثر مقادیر)، این می‌تواند منجر به نشت اطلاعات (data leakage) شود.

۲. **اسکیلر فقط بر اساس داده‌های آموزش تنظیم می‌شود** :نرمال‌سازی داده‌های آزمون باید با استفاده از مقادیر (مانند حداقل و حداکثر) محاسبه‌شده از داده‌های آموزش انجام شود.

```
3. from sklearn.preprocessing import MinMaxScaler
4. scaler = MinMaxScaler()
5.
6. X_train_normalized =
   pd.DataFrame(scaler.fit_transform(X_train),
   columns=X_train.columns)
7.
8. X_test_normalized = pd.DataFrame(scaler.transform(X_test),
   columns=X_test.columns)
9.
10.      print("Normalized Training Data (First 5 Rows):")
11.      print(X_train_normalized.head())
12.
13.      print("\nNormalized Test Data (First 5 Rows):")
14.      print(X_test_normalized.head())
```

نکته:

برای نرمال‌سازی داده‌های آزمون، از مقادیر حداقل و حداکثر محاسبه‌شده از داده‌های آموزش استفاده می‌کنیم (scaler.fit) فقط روی X_train اعمال می‌شود.

```
Normalized Training Data (First 5 Rows):
   GRE Score  TOEFL Score  University Rating    SOP   LOR       CGPA  Research
0       0.22     0.500000               0.75  0.375  0.500  0.535256       0.0
1       0.52     0.321429               0.25  0.375  0.250  0.487179       1.0
2       0.34     0.642857               0.75  0.750  0.875  0.503205       0.0
3       0.12     0.107143               0.25  0.500  0.250  0.237179       1.0
4       0.44     0.464286               0.25  0.375  0.500  0.423077       0.0

Normalized Test Data (First 5 Rows):
   GRE Score  TOEFL Score  University Rating    SOP   LOR       CGPA  Research
0       0.22     0.428571               0.50  0.625  0.75  0.423077       1.0
1       0.42     0.357143               0.50  0.875  0.75  0.589744       1.0
2       1.00     0.785714               1.00  0.750  0.75  0.897436       1.0
3       0.70     0.571429               0.75  0.875  0.75  0.724359       1.0
4       0.22     0.178571               0.25  0.500  0.50  0.346154       1.0
```

ج)

٪۱۰ از داده های آموزش را به ارزیابی تقسیم کردم و دو مدل تک لایه پنهانی و ۳ لایه پنهانی ساختم.

در لایه آخر، از تابع sigmoid استفاده کردم زیرا:

$$\sigma(z) = \frac{1}{1+ e^{-z}}$$

تابع سیگموید (Sigmoid) یک تابع ریاضی است که ورودی‌های خود را به بازه [0,1] نگاشت می‌کند. این ویژگی باعث می‌شود که به طور گسترده در مسائل مربوط به پیش‌بینی احتمالات و مدل‌های یادگیری ماشین استفاده شود، به خصوص در لایه‌های خروجی برای طبقه‌بندی دودویی.

بهینه سازAdam :

ویژگی‌ها و دلایل استفاده:

۱. **به‌روزرسانی سریع‌تر و مؤثرتر وزن‌ها:**

- Adamاز میانگین متحرک مرتبه اول و دوم گرادیان‌ها استفاده می‌کند. این کار باعث می‌شود مسیر گرادیان به‌طور هوشمندانه تنظیم شود و یادگیری سریع‌تر به همگرایی برسد.

۲. **انطباقی بودن نرخ یادگیری:**

- Adamبرای هر پارامتر نرخ یادگیری انطباقی تنظیم می‌کند. این ویژگی برای مسائل پیچیده با داده‌های متنوع بسیار مفید است.

۳. **مقاومت در برابر مشکلات گرادیان ناپدیدشونده:**

- Adamمی‌تواند در مسائل یادگیری عمیق که مشکل گرادیان ناپدیدشونده رخ می‌دهد، بهتر عمل کند.

تابع خطا MAE:

۱. **سادگی تفسیر:**

- MAEبه سادگی بیان می‌کند که به‌طور متوسط، مدل چقدر خطا دارد. مقادیر MAE به واحد داده هدف وابسته‌اند و تفسیر ساده‌ای دارند.

۲. **مقاومت در برابر خطاهای بزرگ:**

- در مقایسه با معیارهایی مانند(MSE (Mean Squared Error ، MAEنسبت به مقادیر پرت (Outliers) حساسیت کمتری دارد.

۳. **معیار مستقیم برای خطا:**

- MAEدقیقاً میانگین خطاهای مطلق را اندازه‌گیری می‌کند، که برای بسیاری از مسائل رگرسیون معیار مناسبی است.

این ترکیب به‌طور کلی مناسب مسائل **رگرسیون احتمالی** یا زمانی است که خطای مطلق به جای خطای مربعی اهمیت بیشتری دارد.

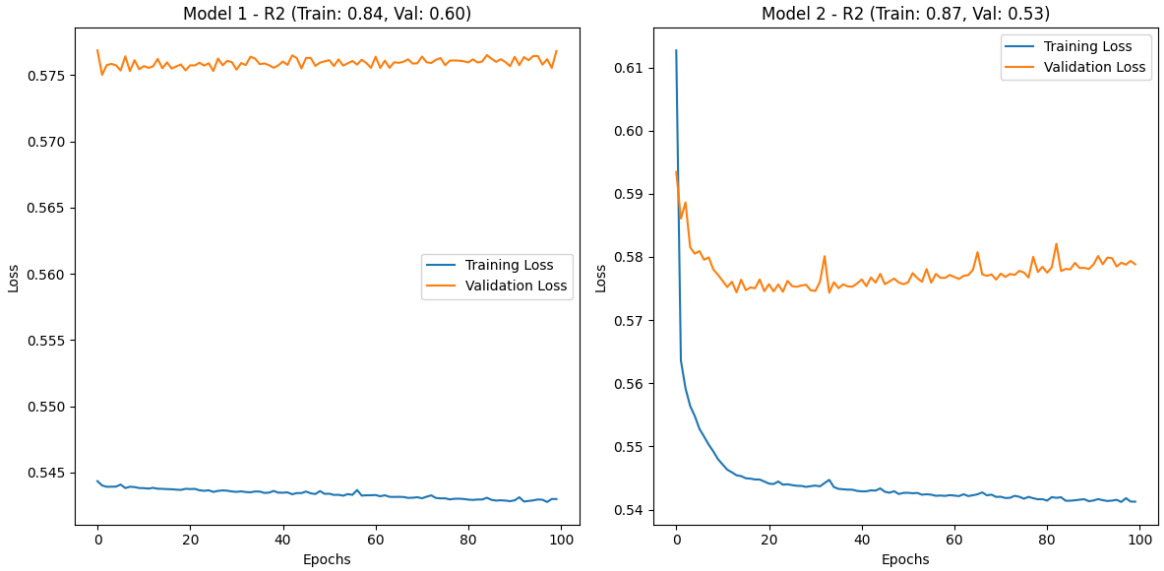
با استفاده از کد زیر مدل را برای ۱۰۰ epochs و بچ ۳۲ آموزش می دهیم:

```
history_1 = model_1.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
    batch_size=32,
    verbose=0
)

history_2 = model_2.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
```

```
    batch_size=32,
    verbose=0
)
```

با محاسبه R2 Score و نمایش آن داریم:



تحلیل مدل‌ها:

مدل ۱:

- **R2 score در داده‌های آموزش: ۰.۸۴**
- **R2 در داده‌های اعتبارسنجی: ۰.۶۰**
- **الگوی اتلاف:**
  - اتلاف در آموزش به تدریج کاهش یافته است و نسبتاً پایدار است.
  - اتلاف اعتبارسنجی در سطح تقریباً ثابتی باقی مانده است.
- **توضیح :**این مدل به نظر می‌رسد که با داده‌های آموزش عملکرد خوبی دارد اما در داده‌های اعتبارسنجی کمی ضعیف عمل کرده است.

مدل ۲:

- **R2در داده‌های آموزش: ۰.۸۷**
- **R2در داده‌های اعتبارسنجی: ۰.۵۳**

- **الگوی اتلاف:**
  - اتلاف در داده‌های آموزش به‌خوبی کاهش یافته است.
  - اما اتلاف اعتبارسنجی مقداری نوسان دارد و از یک نقطه به بعد ثابت نمی‌شود.
- **توضیح :**این مدل ممکن است کمی دچار **Overfitting**شده باشد، زیرا در داده‌های آموزش بهتر عمل می‌کند، اما در اعتبارسنجی R2پایین‌تری نسبت به مدل اول دارد.

انتخاب بهترین مدل:

- **مدل ۱** پایدارتر است و در اعتبارسنجی عملکرد بهتری دارد.
- **مدل ۲** با وجود عملکرد بهتر در آموزش، ممکن است به دلیل پیچیدگی زیاد و نوسانات، به داده‌های آموزش بیش از حد وابسته باشد.

نتیجه‌گیری:

مدل ۱ به دلیل عملکرد بهتر در داده‌های اعتبارسنجی و نداشتن مشکلOverfitting ، گزینه مناسب‌تری است.

مدل یک با کد زیر ذخیره شد.

```
if val_r2_1 > val_r2_2:
    model_1.save('best_model.h5')
    print("Model 1 saved as the best model.")
else:
    model_2.save('best_model.h5')
    print("Model 2 saved as the best model.")
```

د)

مدل بسیار عالی کار نمیکند اما نتایج بدی هم ندارد.

Predicted vs Actual Chance of Admit for 5 Random Samples:

	Actual Chance of Admit	Predicted Chance of Admit
0	0.45	74.939036
1	0.71	69.047642
2	0.71	73.155993
3	0.56	74.918598
4	0.74	77.294284

دلایل احتمالی:

۱. **اشباع تابع سیگموید:**

- اگر وزن‌ها یا گرادیان‌ها بسیار بزرگ شوند، خروجی تابع سیگموید به مقادیر اشباع نزدیک 111یا 000می‌رود.
- این موضوع معمولاً به دلیل داده‌های نامتعادل یا آموزش بیش از حد مدل رخ می‌دهد.

۲. **یادگیری ناکافی از داده‌ها:**

- اگر داده‌ها به درستی نرمال‌سازی یا توزیع نشده باشند، مدل ممکن است نتواند الگوهای مناسبی بیاموزد.

اصلاحات پیشنهادی:

۱. مطمئن شویم داده‌ها به درستی نرمال‌سازی شده‌اند:

اگر داده‌ها به درستی نرمال‌سازی نشده‌اند، مدل ممکن است به وزن‌های بسیار بزرگ برسد.

۲. اضافه کردن Dropout برای جلوگیری از Overfitting:

اگر مدل پیچیدگی زیادی داشته باشد، ممکن است Overfitting رخ دهد.

۳. تنظیم نرخ یادگیری:

نرخ یادگیری را کاهش دهیم تا بهینه‌سازی به‌صورت تدریجی انجام شود.

۵)

(برای جلوگیری از شلوغ شدن گزارش کد را نمیاورم:

ورودی:

```
لطفاً ویژگی‌های زیر را وارد کنید:
GRE Score (بین ۲۶۰ و ۳۴۰): ۲۷۰
TOEFL Score (بین ۸۰ و ۱۲۰): ۹۰
University Rating (بین ۱ و ۵): ۳
SOP (بین ۱ و ۵): ۲
LOR (بین ۱ و ۵): ۵
CGPA (بین ۶,۰ و ۱۰,۰): ۸,۵
Research (بین ۰ و ۱): ۰
```

خروجی:

```
شانس پذیرش شما: ۶۹,۳۰ درصد
```

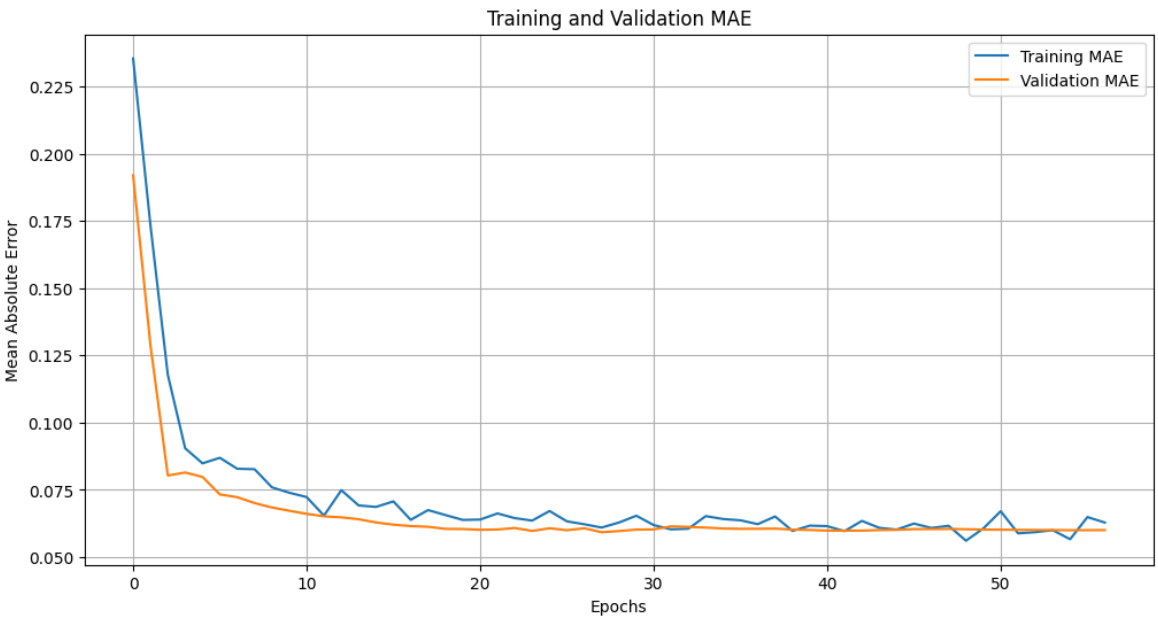
و)

برای هر لایه پنهان Dropout با نرخ بین ۰.۲ تا ۰.۵ اضافه میکنیم. (من ۰.۳ اضافه کردم بر حسب تجربه)

ازReduceLROnPlateau استفاده کردم تا نرخ یادگیری بر اساس عدم بهبود در معیار اعتبارسنجی کاهش یابد.

از EarlyStopping با معیار نظارت (val_loss) و صبر (patience) برابر ۱۰ یا ۱۵ استفاده کنید.

۱. **Dropout**:
   - نرخ ۰.۳ برای لایه‌های پنهان انتخاب شده است.
   - به جلوگیری از وابستگی مدل به نرون‌های خاص کمک می‌کند.
۲. **EarlyStopping**:
   - اگر val_lossبرای ۱۵ دوره پیاپی بهبود نیابد، آموزش متوقف می‌شود.
   - بهترین وزن‌ها پس از توقف بازگردانده می‌شوند.(restore_best_weights=True)
۳. **ReduceLROnPlateau**:
   - اگر val_lossبرای ۵ دوره پیاپی بهبود نیابد، نرخ یادگیری به نصف کاهش می‌یابد.
   - این کار به مدل کمک می‌کند در نهایی‌ترین مراحل آموزش، وزن‌ها را بهبود دهد.



این نمودار تغییرات خطای مطلق میانگین (Mean Absolute Error) را در داده‌های **آموزش (Training)** و **اعتبارسنجی (Validation)** در طول ۵۰ ایپاک اول نمایش می‌دهد. اما ما ۱۰۰۰ ایپاک را تنظیم کرده بودیم. این بدان معناست که احتمالاً **توقف زودهنگام (Early Stopping)** باعث شده است که آموزش در حدود ۵۰ ایپاک متوقف شود.

تمامی پارامترهای مذکور، باعث جلوگیری از بیش برازش خواهند شد.(حتی در مدل های پیچیده)