گزارش میانترم درس مبانی سیستم‌های هوشمند

نام استاد: استاد مهدی علیاری

نگارنده: مهدی خلیل زاده 9932213 – لینک گوگل کولب

---

1 پرسش یک، سوالات هماهنگ شده

آ) الف) رابطه Sensitivity یا حساسیت:

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

رابطه Specificity یا حساسیت:

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

که در آن:

TP: True Positiveموارد به درستی طبقه‌بندی‌شده به عنوان کلاس هدف

TN: True Negativeمواردی که متعلق به کلاس هدف نیستند و به درستی طبقه‌بندی شده‌اند

FP: False Positiveمواردی که به اشتباه به عنوان کلاس هدف طبقه‌بندی شده‌اند

FN: False Negativeمواردی که متعلق به کلاس هدف هستند اما به اشتباه طبقه‌بندی شده‌اند

آ) ب)

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	0.882	0.947
C2	0.744	0.932
C3	0.533	60.96
C4	0.909	30.86

ب)

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 2 نورون است، لایه پنهانی اول دارای 3 نورون، لایه پنهانی دوم دارای 2 نورون و لایه نهایی(خروجی) 1 نورون دارد.

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

فرض 1- وزن‌ها (w) و بایاس‌ها (b) برای هر نورون با مقادیر فرضی مقداردهی می‌شوند.

فرض 2- از تابع فعالسازی سیگموید استفاده می‌کنیم:

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

فرض 3- نرخ یادگیری اتا مشخص شده است.

فرض 4- از تابع میانگین خطای مربع (MSE) استفاده می کنیم:

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

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:

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

$$(z) = 0\sigma$$

Loss Calculation:

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

Backward Pass:

- محاسبه خطای نورون خروجی:

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

- 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:

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

---

2 پرسش دو

آ) شبکه هافیلد یک نوع شبکه عصبی ساده و ابتدایی است که برای ذخیره و بازیابی الگوها طراحی شده است. این شبکه مانند یک حافظه عمل می‌کند که اگر یک الگوی کامل یا حتی ناقص به آن داده شود، می‌تواند الگوی اصلی را بازسازی کند. به زبان ساده، شبکه هافیلد می‌تواند اطلاعاتی که در آن ذخیره شده را به خاطر بیاورد. این شبکه fully connected است و وزن ها تقارن دارند همچنین این شبکه برای رسین به پایداری از یک تابع انرژی استفاده می‌کند که به صورت زیر تعریف می‌شود:

$$E = -0.5\sum_{i}\sum_{j} w_{ij}x_i x_j + \sum_{i}\theta_i x_i$$

در شبکه‌های هافیلد، اگر نشانه‌های گره‌ها (علامت‌های مثبت و منفی) معکوس شوند، الگوی جدید همچنان همان الگوی اصلی باقی می‌ماند. دلایل این امر عبارت‌اند از:

1. تقارن ماتریس وزن‌ها:

در شبکه هافیلد ماتریس وزن ها متقارن هستند، یعنی:

$$w_{ij} = w_{ji}$$

این تقارن باعث می‌شود که مجموع انرژی شبکه با معکوس شدن نشانه گره‌ها تغییر نکند چون در تابع انرژی شبکه که بالاتر آمد، ضرب x_i در x_j ضرب دو عدد اگر علامت هر دو معکوس شود تاثیری روی نتیجه ندارد.

$$x_j x_i = -x_i * -x_j$$

2. خاصیت انرژی مینیمم

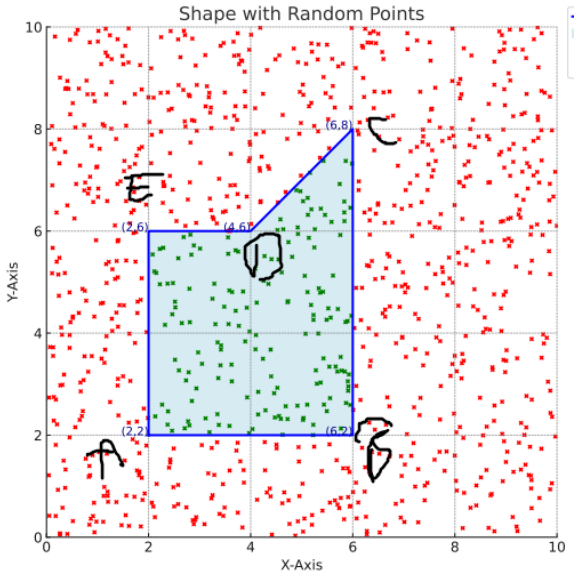
همانطور که ذکر شد این شبکه بر اساس بهینه کردن تابع انرژی و کمینه کردن آن عمل می‌کند و بر این اساس پایدار می‌شود. وقتی نشانه‌های گره‌ها معکوس می‌شود، چون روابط نسبی بین گره‌ها $(x_j x_i)$ ثابت می‌ماند و مقدار انرژی تغییری نمی‌کند $(E = E')$ پس شبکه همچنان در همان حالت پایدار باقی می‌ماند و الگوی ذخیره‌شده تغییری نمی‌کند.

---

3 پرسش سه

آ) برای طراحی شبکه ابتدا باید معادلات مرزی شکل را پیدا کنیمو سپس شبکه عصبی MLP را طراحی کنیم.

نقاط مرزی داده شده:



A: (2, 2) , B: (6, 2), C: (6, 8), D: (4, 6), E: (2, 6)

خط AE:

X = 2:

خطED:

Y = 6

خطDC :

$$m = \frac{6-8}{4-6} = 1 \rightarrow y = x + 2$$

خط CE :

X = 6

خط EA:

Y = 2

ساختار MLP مورد نظر به صورت زیر است:

ورودی: مختصات x, y

لایه پنهان اول: 5 نورون

در این لایه ورودی‌های x, y برای محاسبه فاصله از این خطوط استفاده می‌شوند.

نورون 1: بررسی x = 2

نورون 2: بررسی y = 6

نورون 3: بررسی y = x+2

نورون 4: بررسی x = 6

نورون 5: بررسی y = 2

لایه پنهان دوم:2 نورون

خروجی‌های لایه اول را ترکیب کرده و نقاط را به دو منطقه تقسیم می‌کند (بررسی اشتراک محدودیت‌ها):

- نورون 1 (نورون لایه دوم): بررسی شرایط مرزی مربوط به نقاط در سمت چپ یا پایین شکل
- نورون 2 (نورون لایه دوم): بررسی شرایط مرزی مربوط به نقاط در سمت راست یا بالا شکل

لایه خروجی: 1 نورون برای طبقه بندی نقاط به داخل یا بیرون شبکه که نتیجه نهایی را ارائه می‌دهد.

وزن‌ها و بایاس های لایه اول:

N1: $z_1 = w_{11}x + b_1 \; and \; w_{11} = 1, \; b_1 = -2$

N2: $z_2 = w_{21}y + b_2 \; and \; w_{21} = 1, \; b_2 = -6$

N3: $z_3 = w_{31}x + w_{32}y + b_3 \; and \; w_{31} = -1,\; w_{32} = 1,\; b_3 = -2$

N4: $z_4 = w_{41}x + b_4 \; and \; w_{41} = 1,\; b_4 = -6$

N5: $z_5 = w_{51}x + b_5 \; and \; w_{51} = 1,\; b_5 = -2$

در لایه پنهان دوم هم ترکیب نتایج نورون‌های لایه اول با وزن‌های مناسب برای تعیین این‌که نقطه داخل شکل است یا خیر انجام می‌شود. در لایه خرجی هم یک نورون با تابع فعال‌سازی سیگموید برای خروجی نهایی قرار دارد.

ب) برای مدل سازی شبکه طراحی شده، به صورت oop اول باید ماژول نورون McCulloch Pitts را تعریف کنیم:

```python
# McCulloch Pitts neuron class
class McCulloch_Pitts_neuron:
    def __init__(self, weights, threshold):
        self.weights = np.array(weights)
        self.threshold = threshold

    def model(self, x):
        if np.dot(self.weights, x) >= self.threshold:
            return 1
        else:
            return 0
```

حالا با نورونی که مدل شد کلاس شبکه MLP را مدل می‌کنیم:

```
# Define the MLP model
def MLP(x, y):
    # First layer neurons
    neur1 = McCulloch_Pitts_neuron([1, 0], 2)   # Neuron for x = 2
    neur2 = McCulloch_Pitts_neuron([0, 1], 6)   # Neuron for y = 6
    neur3 = McCulloch_Pitts_neuron([-1, 1], -2) # Neuron for y = x + 2
    neur4 = McCulloch_Pitts_neuron([1, 0], 6)   # Neuron for x = 6
    neur5 = McCulloch_Pitts_neuron([0, 1], 2)   # Neuron for y = 2

    # Outputs from the first layer
    z1 = neur1.model(np.array([x, y]))
    z2 = neur2.model(np.array([x, y]))
    z3 = neur3.model(np.array([x, y]))
    z4 = neur4.model(np.array([x, y]))
    z5 = neur5.model(np.array([x, y]))

    # Second layer neurons
    neur6 = McCulloch_Pitts_neuron([1, 1, 1, 0, 0], 3)  # Combine z1, z2, z3
    neur7 = McCulloch_Pitts_neuron([0, 0, 0, 1, 1], 2)  # Combine z4, z5

    z6 = neur6.model(np.array([z1, z2, z3, z4, z5]))
    z7 = neur7.model(np.array([z1, z2, z3, z4, z5]))

    # Output layer
    neur8 = McCulloch_Pitts_neuron([1, 1], 2)  # Combine z6, z7
    final_output = neur8.model(np.array([z6, z7]))

    return final_output
```

نورون 6 بررسی می‌کند که آیا سه شرط اول (مربوط به خطوط مرزی اول، دوم و سوم) برقرار است یا نه و نورون 7 بررسی می‌کند که آیا دو شرط آخر (مربوط به خطوط مرزی چهارم و پنجم) برقرار است یا نه.

ج) این کار توسط کد زیر انجام شد:

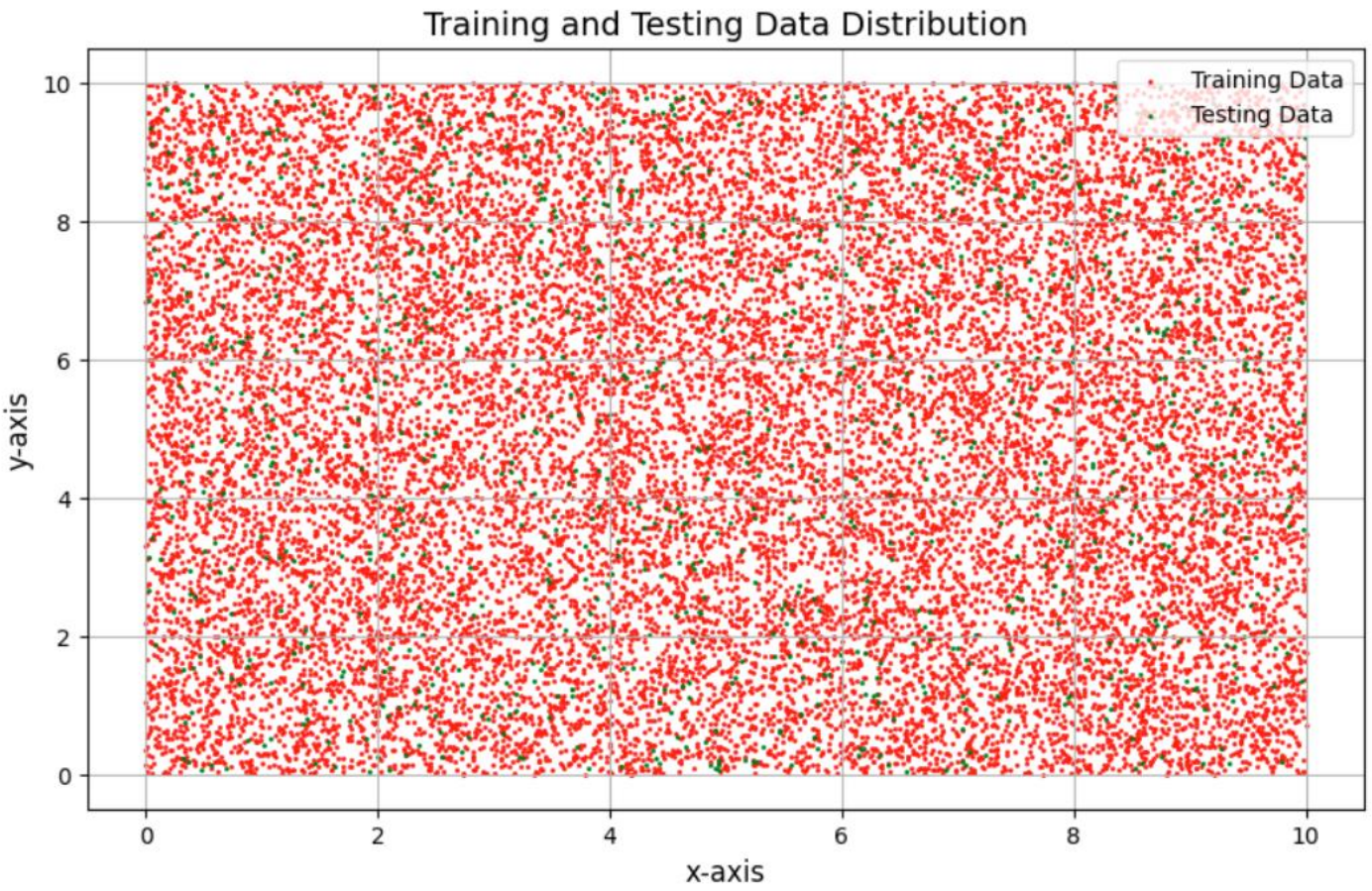
ابتدا یک رندوم سید که دو رقم شماره دانشجوییم بود اتخاب شد تا کد قابل تکرار باشد.بعد با تابع رندوم numpy داده های train و test تولید شدند. سپس با دیتا فریم پاندا در یک csv ذخیره شدند.

```python
# Generate training and testing datasets
np.random.seed(13)  # For reproducibility

# Parameters
n_train = 20000
n_test = 1000
x_min, x_max = 0, 10
y_min, y_max = 0, 10

# Generate random points within the specified range for training
train_x = np.random.uniform(x_min, x_max, n_train)
train_y = np.random.uniform(y_min, y_max, n_train)

# Generate random points within the specified range for testing
test_x = np.random.uniform(x_min, x_max, n_test)
test_y = np.random.uniform(y_min, y_max, n_test)

# Combine into datasets
train_data = pd.DataFrame({'x': train_x, 'y': train_y})
test_data = pd.DataFrame({'x': test_x, 'y': test_y})

# Save datasets to CSV files if needed
train_data.to_csv('training_data.csv', index=False)
test_data.to_csv('testing_data.csv', index=False)
```

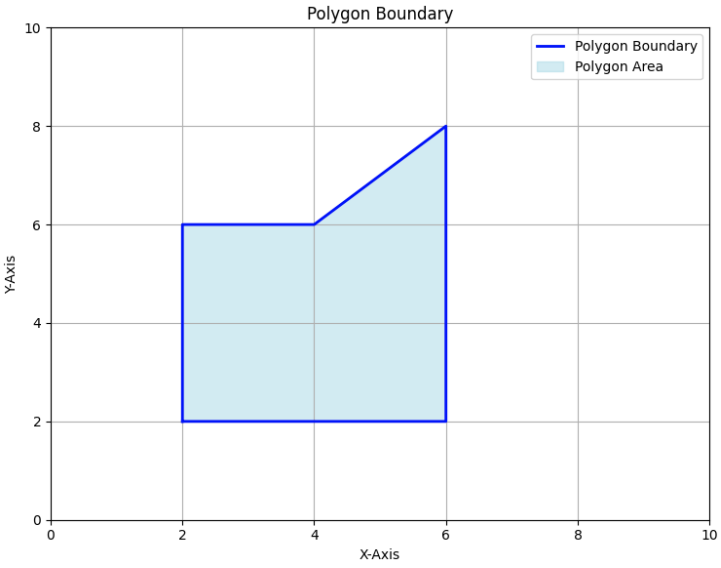
برای بررسی دیتاهای تولید شده با کد زیر پلات شدند:

```python
# Plot the training and testing data
plt.figure(figsize=(10, 6))

# Training data in red
plt.scatter(train_data['x'], train_data['y'], color='red', s=1, label='Training Data')

# Testing data in green
plt.scatter(test_data['x'], test_data['y'], color='green', s=1, label='Testing Data')

# Formatting the plot
plt.title('Training and Testing Data Distribution', fontsize=14)
plt.xlabel('x-axis', fontsize=12)
plt.ylabel('y-axis', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

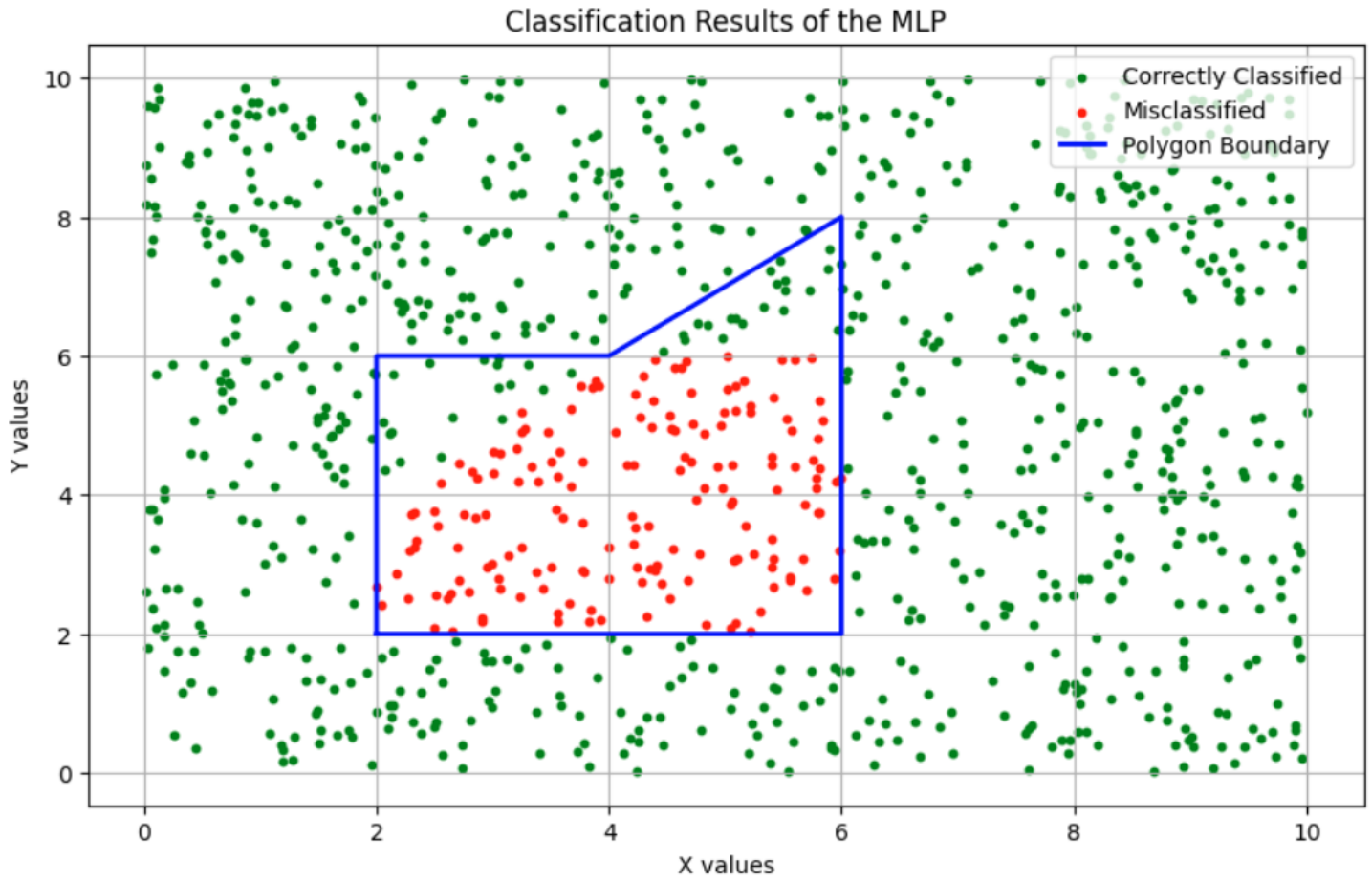
داده ها (سبز برای داده test و قرمز برای داده train)



د) نتیجه به شکل زیر شد که متاسفانه نمیدونم کجا سوتی دادم و درست کار نمیکند.



ه) نتیجه به صورت زیر شد:



4 پرسش چهار

به این صورت دیتا لود شد:

```
# link from google drive: https://drive.google.com/file/d/1wRR_5T_RZzNkpRiv3sqPbg69iwSwWyGH/view?usp=sharing

file_id = "1wRR_5T_RZzNkpRiv3sqPbg69iwSwWyGH"
url = f"https://drive.google.com/uc?id={file_id}"

# File Direction
File_dir = "/content/Admission_Predict.csv"
gdown.download(url, File_dir, quiet=False)
# move to dataframe
df = pd.read_csv('Admission_Predict.csv')
df.head()
```

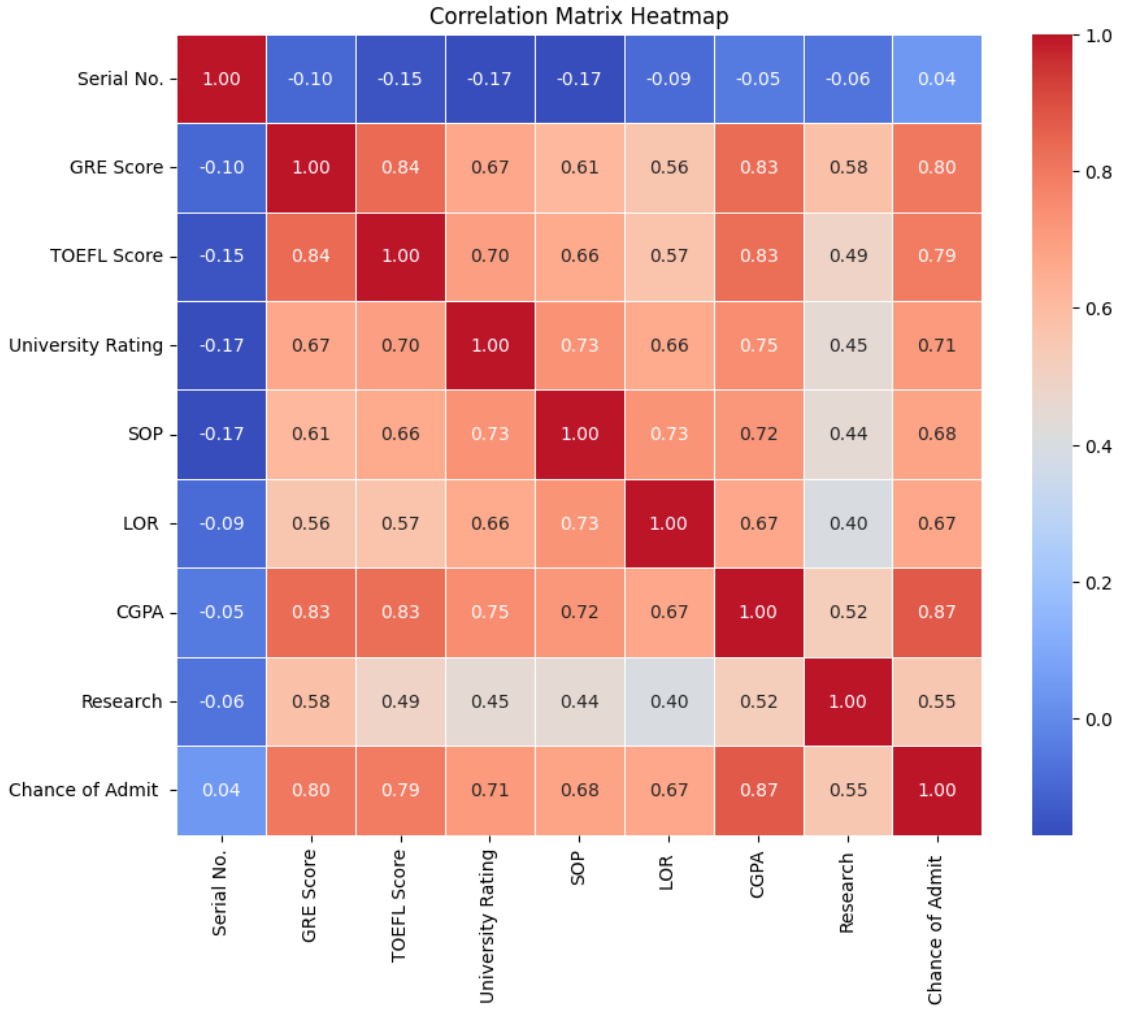
```
Downloading...
From: https://drive.google.com/uc?id=1wRR_5T_RZzNkpRiv3sqPbg69iwSwWyGH
To: /content/Admission_Predict.csv
100%|██████████| 12.9k/12.9k [00:00<00:00, 3.35MB/s]
```

	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

آ) کد مربوطه:

```
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()
```

نتیجه:



ب) داده ا به صئرت زیر تقسیم بندی شدند:

```
y = df["Chance of Admit "]

target_analysis = y.value_counts()
# Splitting data
from sklearn.model_selection import train_test_split

X = df.drop("Chance of Admit ", axis=1)
y = df["Chance of Admit "]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=42)

print("Training Set Size:", X_train.shape)
print("Test Set Size:", X_test.shape)
```

```
Training Set Size: (340, 8)
Test Set Size: (60, 8)
```

چون داده ها مقیاس های متفاوت دارند از نرمال سازی برای پیش پردازش استفاده شد

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X_train_normalized = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test_normalized = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)

print("Normalized Training Data (First 5 Rows):")
print(X_train_normalized.head())

print("\nNormalized Test Data (First 5 Rows):")
print(X_test_normalized.head())
```

```
Normalized Training Data (First 5 Rows):
   Serial No.  GRE Score  TOEFL Score  University Rating    SOP   LOR   \
0    0.309045       0.22     0.500000               0.75  0.375  0.500
1    0.736181       0.52     0.321429               0.25  0.375  0.250
2    0.638191       0.34     0.642857               0.75  0.750  0.875
3    0.193467       0.12     0.107143               0.25  0.500  0.250
4    0.251256       0.44     0.464286               0.25  0.375  0.500

       CGPA  Research
0  0.535256       0.0
1  0.487179       1.0
2  0.503205       0.0
3  0.237179       1.0
4  0.423077       0.0

Normalized Test Data (First 5 Rows):
   Serial No.  GRE Score  TOEFL Score  University Rating    SOP   LOR   \
0    0.522613       0.22     0.428571               0.50  0.625  0.75
1    0.701005       0.42     0.357143               0.50  0.875  0.75
2    0.080402       1.00     0.785714               1.00  0.750  0.75
3    0.525126       0.70     0.571429               0.75  0.875  0.75
4    0.231156       0.22     0.178571               0.25  0.500  0.50

       CGPA  Research
0  0.423077       1.0
1  0.589744       1.0
2  0.897436       1.0
3  0.724359       1.0
4  0.346154       1.0
```