education-linear-regression-akash

December 4, 2023

1 Business Case:Jamboree Education - Linear Regression

About Jamboree

Jamboree has helped thousands of students like you make it to top colleges abroad. Be it GMAT, GRE or SAT, their unique problem-solving methods ensure maximum scores with minimum effort. They recently launched a feature where students/learners can come to their website and check their probability of getting into the IVY league college. This feature estimates the chances of graduate admission from an Indian perspective.

Understanding what factors are important in graduate admissions and how these factors are interrelated among themselves. It will also help predict one's chances of admission given the rest of the variables.

[106]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[6]:
```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/
↪original/Jamboree_Admission.csv
```

```
--2023-12-04 16:29:36--  https://d2beiqkhq929f0.cloudfront.net/public_assets/ass
ets/000/001/839/original/Jamboree_Admission.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)…
65.8.234.72, 65.8.234.36, 65.8.234.131, …
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|65.8.234.72|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 16176 (16K) [text/plain]
Saving to: 'Jamboree_Admission.csv.1'

Jamboree_Admission. 100%[===================>]  15.80K  --.-KB/s    in 0s
```

```
2023-12-04 16:29:36 (229 MB/s) - ‘Jamboree_Admission.csv.1’ saved [16176/16176]
```

[7]:
```python
df=pd.read_csv("Jamboree_Admission.csv")
df.head(10)
```

[7]:
```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0           1        337          118                  4  4.5   4.5  9.65
1           2        324          107                  4  4.0   4.5  8.87
2           3        316          104                  3  3.0   3.5  8.00
3           4        322          110                  3  3.5   2.5  8.67
4           5        314          103                  2  2.0   3.0  8.21
5           6        330          115                  5  4.5   3.0  9.34
6           7        321          109                  3  3.0   4.0  8.20
7           8        308          101                  2  3.0   4.0  7.90
8           9        302          102                  1  2.0   1.5  8.00
9          10        323          108                  3  3.5   3.0  8.60

   Research  Chance of Admit
0         1             0.92
1         1             0.76
2         1             0.72
3         1             0.80
4         0             0.65
5         1             0.90
6         1             0.75
7         0             0.68
8         0             0.50
9         0             0.45
```

1.1 1. Exploratory Data Analysis

[8]:
```python
df=df.drop('Serial No.',axis=1)
```

[9]:
```python
df.head()
```

[9]:
```
   GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research  \
0        337          118                  4  4.5   4.5  9.65         1
1        324          107                  4  4.0   4.5  8.87         1
2        316          104                  3  3.0   3.5  8.00         1
3        322          110                  3  3.5   2.5  8.67         1
4        314          103                  2  2.0   3.0  8.21         0

   Chance of Admit
0             0.92
1             0.76
2             0.72
```

```
3              0.80
4              0.65
```

[10]:
```
df.columns
```

[10]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

[11]:
```
df.shape
```

[11]:
```
(500, 8)
```

[12]:
```
df.isnull().sum() # No missing values good to go
```

[12]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

[13]:
```
df.info() # all datatypes are correctly identified good to go
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          500 non-null    int64
 1   TOEFL Score        500 non-null    int64
 2   University Rating  500 non-null    int64
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

[14]:
```
df.describe()
```

[14]:
```
        GRE Score  TOEFL Score  University Rating         SOP        LOR   \
count  500.000000   500.000000         500.000000  500.000000  500.00000
```

```
mean    316.472000   107.192000           3.114000    3.374000    3.48400
std      11.295148     6.081868           1.143512    0.991004    0.92545
min     290.000000    92.000000           1.000000    1.000000    1.00000
25%     308.000000   103.000000           2.000000    2.500000    3.00000
50%     317.000000   107.000000           3.000000    3.500000    3.50000
75%     325.000000   112.000000           4.000000    4.000000    4.00000
max     340.000000   120.000000           5.000000    5.000000    5.00000

             CGPA    Research  Chance of Admit
count  500.000000  500.000000        500.00000
mean     8.576440    0.560000          0.72174
std      0.604813    0.496884          0.14114
min      6.800000    0.000000          0.34000
25%      8.127500    0.000000          0.63000
50%      8.560000    1.000000          0.72000
75%      9.040000    1.000000          0.82000
max      9.920000    1.000000          0.97000
```

[108]:

```
df.rename(columns={'LOR ':'LOR', 'Chance of Admit ':'Chance of Admit'},␣
↪inplace=True)
```

[99]:

```
# Duplicate values in the dataset
df.duplicated().sum()
```

[99]: 0

[100]:

```
# unique values in the dataset
for col in df:
    print(f'Number of unique values in the {col} column:',df[col].nunique())
```

```
Number of unique values in the Serial No. column: 500
Number of unique values in the GRE Score column: 49
Number of unique values in the TOEFL Score column: 29
Number of unique values in the University Rating column: 5
Number of unique values in the SOP column: 9
Number of unique values in the LOR  column: 9
Number of unique values in the CGPA column: 184
Number of unique values in the Research column: 2
Number of unique values in the Chance of Admit  column: 61
Number of unique values in the ratio_CGPA_GRE column: 468
Number of unique values in the ratio_CGPA_TOEFL column: 435
Number of unique values in the Chance of Admit column: 61
```

[101]:

```
for i in df.columns:
    print(i, '--> ','\n', df[i].unique(), '\n')
```

```
Serial No. -->
 [  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
```

```
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
 487 488 489 490 491 492 493 494 495 496 497 498 499 500]

GRE Score -->
 [337 324 316 322 314 330 321 308 302 323 325 327 328 307 311 317 319 318
 303 312 334 336 340 298 295 310 300 338 331 320 299 304 313 332 326 329
 339 309 315 301 296 294 306 305 290 335 333 297 293]

TOEFL Score -->
 [118 107 104 110 103 115 109 101 102 108 106 111 112 105 114 116 119 120
  98  93  99  97 117 113 100  95  96  94  92]

University Rating -->
 [4 3 2 5 1]

SOP -->
 [4.5 4.  3.  3.5 2.  5.  1.5 1.  2.5]

LOR  -->
 [4.5 3.5 2.5 3.  4.  1.5 2.  5.  1. ]

CGPA -->
 [9.65 8.87 8.   8.67 8.21 9.34 8.2  7.9  8.6  8.4  9.   9.1  8.3  8.7
```

```
 8.8  8.5  9.5  9.7  9.8  9.6  7.5  7.2  7.3  8.1  9.4  9.2  7.8  7.7
 9.3  8.85 7.4  7.6  6.8  8.92 9.02 8.64 9.22 9.16 9.64 9.76 9.45 9.04
 8.9  8.56 8.72 8.22 7.54 7.36 8.02 9.36 8.66 8.42 8.28 8.14 8.76 7.92
 7.66 8.03 7.88 7.84 8.96 9.24 8.88 8.46 8.12 8.25 8.47 9.05 8.78 9.18
 9.46 9.38 8.48 8.68 8.34 8.45 8.62 7.46 7.28 8.84 9.56 9.48 8.36 9.32
 8.71 9.35 8.65 9.28 8.77 8.16 9.08 9.12 9.15 9.44 9.92 9.11 8.26 9.43
 9.06 8.75 8.89 8.69 7.86 9.01 8.97 8.33 8.27 7.98 8.04 9.07 9.13 9.23
 8.32 8.98 8.94 9.53 8.52 8.43 8.54 9.91 9.87 7.65 7.89 9.14 9.66 9.78
 9.42 9.26 8.79 8.23 8.53 8.07 9.31 9.17 9.19 8.37 7.68 8.15 8.73 8.83
 8.57 9.68 8.09 8.17 7.64 8.01 7.95 8.49 7.87 7.97 8.18 8.55 8.74 8.13
 8.44 9.47 8.24 7.34 7.43 7.25 8.06 7.67 9.54 9.62 7.56 9.74 9.82 7.96
 7.45 7.94 8.35 7.42 8.95 9.86 7.23 7.79 9.25 9.67 8.86 7.57 7.21 9.27
 7.81 7.69]

Research -->
 [1 0]

Chance of Admit  -->
 [0.92 0.76 0.72 0.8  0.65 0.9  0.75 0.68 0.5  0.45 0.52 0.84 0.78 0.62
 0.61 0.54 0.66 0.63 0.64 0.7  0.94 0.95 0.97 0.44 0.46 0.74 0.91 0.88
 0.58 0.48 0.49 0.53 0.87 0.86 0.89 0.82 0.56 0.36 0.42 0.47 0.55 0.57
 0.96 0.93 0.38 0.34 0.79 0.71 0.69 0.59 0.85 0.77 0.81 0.83 0.67 0.73
 0.6  0.43 0.51 0.39 0.37]

ratio_CGPA_GRE -->
 [2.86350148 2.73765432 2.53164557 2.69254658 2.61464968 2.83030303
 2.55451713 2.56493506 2.64900662 2.6625387  2.58461538 2.75229358
 2.77439024 2.60586319 2.63665595 2.6433121  2.7444795  2.50783699
 2.7672956  2.80528053 2.53205128 2.89634146 2.90419162 2.91666667
 2.82352941 2.73291925 2.51677852 2.44067797 2.35483871 2.7
 2.5382263  2.78106509 2.96072508 2.875      2.80936455 2.6
 2.46710526 2.50814332 2.5974026  2.59493671 2.71565495 2.74096386
 2.88343558 2.82608696 2.82674772 2.86135693 2.75700935 2.56880734
 2.65175719 2.39520958 2.5        2.48447205 2.40625    2.34177215
 2.55033557 2.26666667 2.66881029 2.62135922 2.67100977 2.69736842
 2.6984127  2.67692308 2.74461538 2.75840979 2.73417722 2.89937107
 2.79268293 2.90361446 2.9047619  2.94392523 2.87898089 2.8343949
 2.60182371 2.66666667 2.73089701 2.5472973  2.50340136 2.57051282
 2.79411765 2.88125    2.9068323  2.77941176 2.71473354 2.67301587
 2.61198738 2.59235669 2.7721519  2.49056604 2.56187291 2.69463087
 2.6179402  2.52805281 2.57894737 2.61437908 2.70694864 2.78313253
 2.74922601 2.62732919 2.6025641  2.62738854 2.67192429 2.77607362
 2.77848101 2.79027356 2.79881657 2.83383686 2.84210526 2.78032787
 2.70404984 2.77076412 2.675      2.7170418  2.91612903 2.88294314
 2.57241379 2.45945946 2.70336391 2.85373134 2.83832335 2.69677419
 2.66883117 2.81395349 2.88666667 2.88544892 2.73040752 2.79141104
 2.80780781 2.87905605 2.85478548 2.77022654 2.71826625 2.78678679
 2.79299363 2.70833333 2.58227848 2.78527607 2.86792453 2.78115502
```

2.81927711	2.85196375	2.91764706	2.75692308	2.69206349	2.79447853
2.89085546	2.65594855	2.82335329	2.79518072	2.82242991	2.70061728
2.72699387	2.78525641	2.64761905	2.67313916	2.66013072	2.65993266
2.4952381	2.5033557	2.67295597	2.70031546	2.73860182	2.78571429
2.75827815	2.64217252	2.66211604	2.5659164	2.57692308	2.71556886
2.83540373	2.85758514	2.79439252	2.771875	2.78419453	2.8338558
2.62783172	2.69381107	2.72	2.76065574	2.63545151	2.80254777
2.63291139	2.78593272	2.73817035	2.81791045	2.82779456	2.80246914
2.82716049	2.78018576	2.77639752	2.83630952	2.7752443	2.69934641
2.68709677	2.71061093	2.77635783	2.69400631	2.68571429	2.91470588
2.95508982	2.86577181	2.59322034	2.5047619	2.58709677	2.67540984
2.69767442	2.78769231	2.78658537	2.85798817	2.93693694	2.84592145
2.83636364	2.8757764	2.84423676	2.76851852	2.69871795	2.79552716
2.70886076	2.71296296	2.74350649	2.69836066	2.71283784	2.76143791
2.73397436	2.72641509	2.7808642	2.76357827	2.61128527	2.65064103
2.65460526	2.82121212	2.83128834	2.82153846	2.79331307	2.7
2.63879599	2.59459459	2.57097792	2.7037037	2.78153846	2.72611465
2.75	2.76265823	2.7266881	2.75077882	2.678125	2.84810127
2.68553459	2.88955224	2.8411215	2.72638436	2.68711656	2.82175227
2.79204893	2.59294872	2.71428571	2.69538462	2.77316294	2.60191083
2.79510703	2.70779221	2.68627451	2.62876254	2.59863946	2.56730769
2.52380952	2.7826087	2.87234043	2.69375	2.75649351	2.87171053
2.77813505	2.87381703	2.81730769	2.77258567	2.84117647	2.79758308
2.73511905	2.87261146	2.87539936	2.49185668	2.62333333	2.63907285
2.62179487	2.70347003	2.79677419	2.846875	2.8	2.83606557
2.58899676	2.7460815	2.6242236	2.64705882	2.69329073	2.74143302
2.81733746	2.76923077	2.79220779	2.73125	2.79878049	2.89389068
2.67109635	2.66557377	2.62012987	2.63758389	2.67	2.71604938
2.65749235	2.68138801	2.6130031	2.63375796	2.68196721	2.64444444
2.80368098	2.68227425	2.66440678	2.70679012	2.65656566	2.64831804
2.61093248	2.66558442	2.67711599	2.7724359	2.80307692	2.75548589
2.85240964	2.70588235	2.67283951	2.71153846	2.67532468	2.48813559
2.35126582	2.51315789	2.4548495	2.40066225	2.5686901	2.60062893
2.66769231	2.66006601	2.72333333	2.58249158	2.5615142	2.68195719
2.62126246	2.43312102	2.62928349	2.68322981	2.85628743	2.73076923
2.73202614	2.84345048	2.77878788	2.60625	2.39871383	2.6442953
2.66777409	2.65806452	2.84567901	2.86309524	2.66043614	2.42857143
2.51973684	2.5016835	2.60689655	2.52475248	2.74534161	2.71786834
2.82407407	2.75333333	2.86470588	2.93134328	2.63576159	2.63843648
2.63513514	2.6375	2.62420382	2.72012579	2.79754601	2.76340694
2.80547112	2.79012346	2.86363636	2.81410256	2.9009009	2.70394737
2.75925926	2.7969697	2.45659164	2.46688742	2.49068323	2.66778523
2.67333333	2.51162791	2.59744409	2.50955414	2.50473186	2.60124611
2.67584098	2.57142857	2.43037975	2.5987055	2.59090909	2.48160535
2.78816199	2.80124224	2.91076923	3.05263158	2.74679487	2.79032258
2.76582278	2.66470588	2.61414791	2.74375	2.70253165	2.68300654
2.48543689	2.33225806	2.41324921	2.55409836	2.60843373	2.85358255
2.85493827	2.76829268	2.95718654	2.803125	2.89102564	2.8984127

```
 2.76875    2.92682927 2.58387097 2.50491803 2.56610169 2.8449848
 2.70099668 2.58631922 2.58552632 2.41946309 2.70819672 2.78025478
 2.76100629 2.80981595 2.896875   2.60128617 2.89908257 2.57911392
 2.58116883 2.62666667 2.67105263 2.64724919 2.66981132 2.80685358
 2.625387   2.67378049 2.60526316 2.48895899 2.68167203 2.62382445
 2.67701863 2.81456954 2.64495114 2.62962963 2.58053691 2.74
 2.80730897 2.71686747 2.92878338 2.8969697  2.70192308 2.76452599]

ratio_CGPA_TOEFL -->
 [8.1779661  8.28971963 7.69230769 7.88181818 7.97087379 8.12173913
 7.52293578 7.82178218 7.84313725 7.96296296 7.9245283  8.10810811
 8.125      7.33944954 7.88461538 7.9047619  8.13084112 7.54716981
 8.         8.33333333 7.38317757 7.36842105 8.18965517 8.1512605
 8.23529412 8.0733945  7.65306122 7.74193548 7.37373737 8.35051546
 8.05825243 7.96610169 8.42105263 8.75       8.36363636 7.42857143
 7.14285714 7.12962963 7.27272727 7.80952381 7.94392523 7.77777778
 8.31858407 8.27272727 8.15789474 8.04545455 7.56756757 8.46938776
 7.9        6.89655172 7.23214286 7.47572816 7.25490196 7.67676768
 6.86868687 7.98076923 8.1        8.11881188 7.83783784 7.96428571
 7.9122807  8.07476636 8.4587156  7.96521739 8.16949153 8.71428571
 8.51351351 8.37037037 8.39622642 7.50877193 7.78571429 8.3030303
 7.93684211 7.91397849 7.63809524 7.91666667 8.38181818 8.13913043
 8.2173913  8.40776699 7.94339623 7.73831776 7.53703704 8.03669725
 7.47169811 7.89690722 8.19387755 8.12371134 7.73737374 7.84
 7.46666667 7.76470588 7.85840708 7.90654206 7.73333333 7.78301887
 8.14423077 8.08035714 7.98181818 8.27027027 8.08547009 8.0862069
 8.38834951 7.85185185 7.96330275 7.79439252 7.78181818 8.04761905
 8.52830189 8.45098039 7.17307692 7.35353535 8.5        8.17094017
 7.96638655 7.88679245 7.61111111 7.99056604 8.66       8.24778761
 7.77678571 7.92372881 8.56140351 8.23809524 8.15238095 7.83928571
 8.21238938 8.04587156 8.2038835  8.16       7.82758621 8.36697248
 8.31818182 7.93220339 8.20869565 8.26666667 7.6460177  8.07619048
 7.99122807 8.44827586 7.79245283 8.27192982 8.         8.08928571
 8.23148148 7.97247706 7.94285714 7.94230769 7.67924528 7.63106796
 7.53535354 7.79816514 8.11711712 8.15454545 8.16666667 8.10784314
 8.04123711 8.06060606 7.96039604 7.75213675 8.3        8.16814159
 8.08108108 7.99099099 7.69747899 8.21818182 7.51851852 7.84615385
 7.86915888 7.88       7.8490566  8.0619469  8.11214953 8.10714286
 8.25225225 8.16363636 7.84210526 8.07627119 7.96261682 7.86666667
 7.85849057 8.10576923 8.12149533 8.29126214 7.69090909 8.25833333
 8.225      8.13333333 7.72727273 7.96969697 7.8627451  7.69811321
 7.80769231 8.38888889 8.30909091 8.05       8.21848739 8.05128205
 8.06896552 8.26785714 8.37614679 8.09615385 8.49514563 7.77876106
 7.75229358 7.83809524 8.11111111 7.68181818 7.75454545 7.74107143
 8.31730769 7.72897196 8.07       8.23893805 8.31531532 8.1875
 8.06140351 8.04807692 7.89       7.6039604  7.91262136 7.6173913
 7.92982456 8.2        8.31428571 8.15384615 8.06363636 7.95495495
 8.24038462 9.09090909 8.54       8.4173913  7.60909091 8.64646465
```

```
 8.64       8.58823529 7.8487395  8.4537037  7.77884615 8.11650485
 7.91891892 7.96363636 8.50980392 7.63551402 8.08849558 7.72222222
 7.82857143 8.04210526 8.09090909 7.95       8.14545455 8.36283186
 8.53465347 8.24271845 8.55882353 8.47058824 8.28181818 8.29245283
 8.01801802 8.625      7.98275862 7.78813559 7.96491228 8.67307692
 8.25688073 7.28571429 7.71568627 8.05050505 8.34693878 8.23762376
 8.57       8.10280374 7.59166667 8.10526316 7.72321429 7.99065421
 7.95283019 8.03571429 7.89814815 7.81818182 8.40384615 8.41121495
 8.04       7.74285714 7.75961538 7.78217822 7.92792793 7.69026549
 8.01886792 8.11538462 8.01960784 8.00961538 7.87931034 8.02
 7.83035714 8.21875    7.66371681 7.74528302 7.90740741 8.08411215
 8.20720721 7.99090909 8.02542373 8.09259259 8.09345794 7.77358491
 7.89320388 7.64583333 7.58163265 7.87628866 7.80851064 7.32323232
 8.06       8.00980392 7.82653061 7.66037736 7.58653846 7.27619048
 7.88785047 7.85454545 8.22413793 8.02608696 8.7254902  8.04385965
 8.01923077 7.6122449  8.56521739 8.08403361 7.83486239 7.58415842
 7.73958333 7.56       7.80612245 8.51515152 8.25714286 8.26
 8.61946903 8.39316239 7.88118812 7.71428571 7.81481481 8.07843137
 8.16037736 8.14285714 8.42307692 8.51401869 8.14655172 8.52427184
 8.25641026 8.22       8.20183486 7.95689655 7.56435644 7.52525253
 7.78640777 7.59405941 8.18367347 7.875      8.64893617 7.7254902
 7.86138614 7.59090909 8.25471698 7.78846154 7.45631068 7.23423423
 7.82352941 7.42       7.99107143 8.05357143 8.01680672 8.29824561
 8.88288288 8.08490566 8.56435644 8.48543689 7.87826087 7.81730769
 8.23214286 7.69369369 7.31428571 6.57272727 7.21698113 7.76363636
 8.05714286 7.49038462 7.73214286 8.13793103 8.03508772 8.18584071
 8.19491525 8.30555556 8.27522936 9.03960396 7.91071429 8.27586207
 8.50485437 7.62857143 7.49019608 7.64646465 8.53       8.28318584
 8.24761905 7.97058824 7.56190476 7.34579439 7.43298969 8.60416667
 8.81818182 8.69306931 8.07272727 8.42727273 7.85436893 8.17241379
 7.99019608 7.57142857 7.8019802  7.79047619 8.83333333 7.92523364
 7.76106195 7.68932039 7.44339623 8.25742574 8.20588235 7.94782609
 7.69642857 7.88888889 7.61386139 8.65263158 8.53535354 8.35185185
 8.43589744 7.96666667 8.18446602]

Chance of Admit -->
 [0.92 0.76 0.72 0.8  0.65 0.9  0.75 0.68 0.5  0.45 0.52 0.84 0.78 0.62
 0.61 0.54 0.66 0.63 0.64 0.7  0.94 0.95 0.97 0.44 0.46 0.74 0.91 0.88
 0.58 0.48 0.49 0.53 0.87 0.86 0.89 0.82 0.56 0.36 0.42 0.47 0.55 0.57
 0.96 0.93 0.38 0.34 0.79 0.71 0.69 0.59 0.85 0.77 0.81 0.83 0.67 0.73
 0.6  0.43 0.51 0.39 0.37]
```

1.1.1 Univariate Analysis

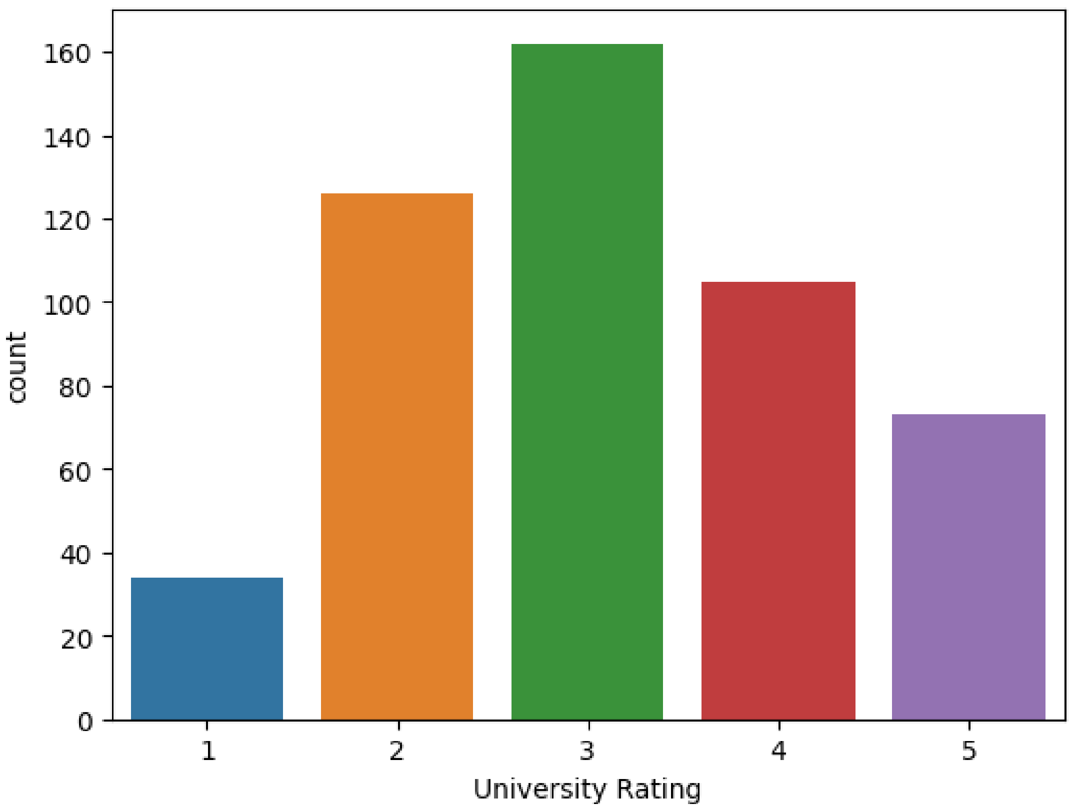
[15]:
```python
df["University Rating"].unique()
```

[15]: array([4, 3, 2, 5, 1])

[16]:
```
df["University Rating"].value_counts() # maximum applicant are from university␣
↪rating 3
```

[16]:
```
3    162
2    126
4    105
5     73
1     34
Name: University Rating, dtype: int64
```

[17]:
```
sns.countplot(data=df,x="University Rating") # Visual representation of the␣
↪above code
plt.show()
```



[18]:
```
df["GRE Score"].unique()
```

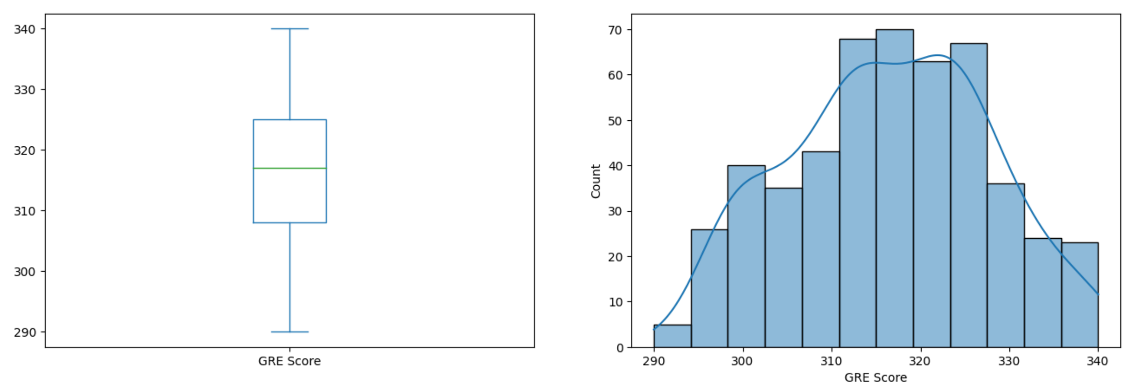
[18]:
```
array([337, 324, 316, 322, 314, 330, 321, 308, 302, 323, 325, 327, 328,
       307, 311, 317, 319, 318, 303, 312, 334, 336, 340, 298, 295, 310,
       300, 338, 331, 320, 299, 304, 313, 332, 326, 329, 339, 309, 315,
```

```
       301, 296, 294, 306, 305, 290, 335, 333, 297, 293])
```

```
[19]: df["GRE Score"].value_counts(bins=5) # maximum applicant score in GRE Score is␣
       ↪lie between 300 to 330 out of 350
```

```
[19]: (310.0, 320.0]      154
      (320.0, 330.0]      141
      (300.0, 310.0]       96
      (330.0, 340.0]       56
      (289.949, 300.0]     53
      Name: GRE Score, dtype: int64
```

```
[20]: plt.subplot(121)
      df["GRE Score"].plot.box(figsize=(16,5))    # Median is at 317
      plt.subplot(122)                            # GRE Score data is  normaly␣
       ↪distributed
      sns.histplot(df["GRE Score"], kde=True)     # no outliers present
      plt.show()
```




```
[21]: df["TOEFL Score"].unique()
```

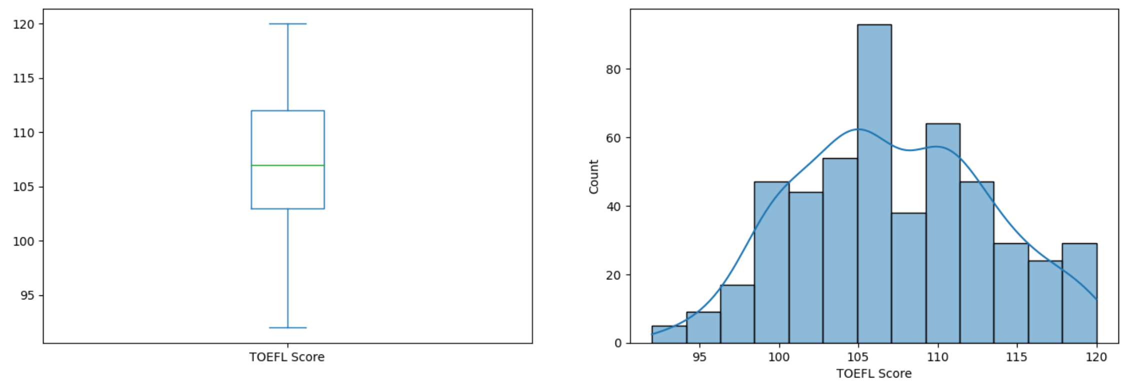
```
[21]: array([118, 107, 104, 110, 103, 115, 109, 101, 102, 108, 106, 111, 112,
             105, 114, 116, 119, 120,  98,  93,  99,  97, 117, 113, 100,  95,
              96,  94,  92])
```

```
[22]: df["TOEFL Score"].value_counts(bins=5) # maximum applicant score in TOEFL Score␣
       ↪is lie between 95 to 120 out of 120
```

```
[22]: (108.8, 114.4]                148
      (103.2, 108.8]                141
      (97.6, 103.2]                 126
      (114.4, 120.0]                 64
      (91.97099999999999, 97.6]      21
```

```
Name: TOEFL Score, dtype: int64
```

[23]:
```
plt.subplot(121)
df["TOEFL Score"].plot.box(figsize=(16,5))     # Median is at 107
plt.subplot(122)                               # TOEFL Score data is  normaly␣
 ↪distributed
sns.histplot(df["TOEFL Score"], kde=True)      # no outliers present
plt.show()
```



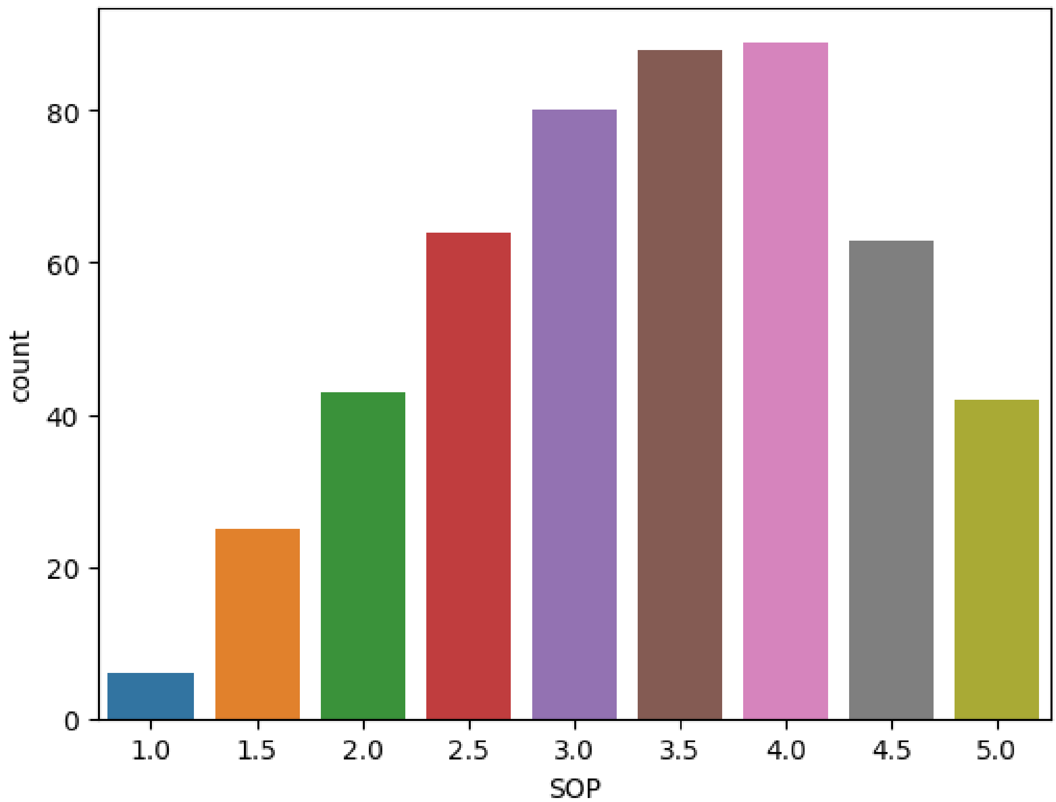
[24]:
```
df["SOP"].unique()
```

[24]:
```
array([4.5, 4. , 3. , 3.5, 2. , 5. , 1.5, 1. , 2.5])
```

[25]:
```
df["SOP"].value_counts(bins=2) # Maximum applicants Statement of Purpose and␣
 ↪Letter of Recommendation Strength lie between 3 to 5 out of 5
```

[25]:
```
(3.0, 5.0]       282
(0.995, 3.0]     218
Name: SOP, dtype: int64
```

[26]:
```
sns.countplot(data=df,x="SOP") # Visual representation of the above code
plt.show()
```



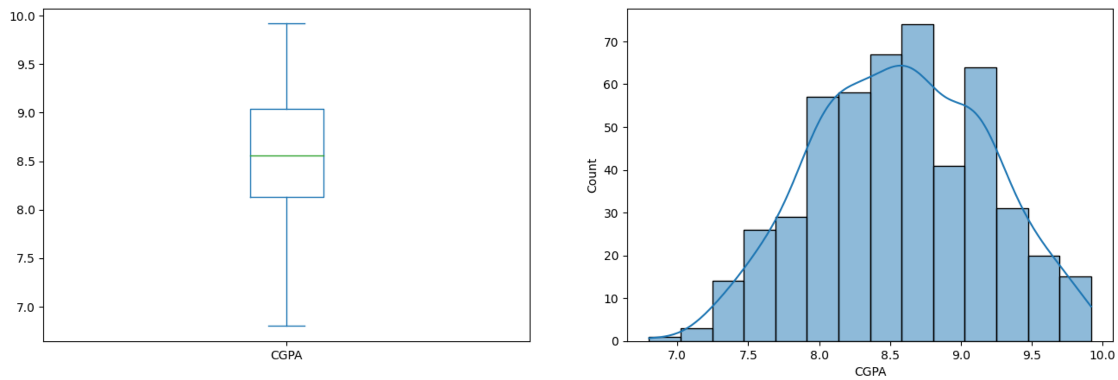
[27]:
```
df["CGPA"].nunique()
```

[27]:
```
184
```

[28]:
```
df["CGPA"].value_counts(bins=5) # Maximum applicants Undergraduate GPA score␣
↪lie between 7 to 9 out of 10
```

[28]:
```
(8.048, 8.672]                 175
(8.672, 9.296]                 156
(7.424, 8.048]                  96
(9.296, 9.92]                   61
(6.795999999999999, 7.424]      12
Name: CGPA, dtype: int64
```

[29]:
```
plt.subplot(121)
df["CGPA"].plot.box(figsize=(16,5))     # Median is at 8.56
plt.subplot(122)                        # CGPA Score data is normaly distributed
sns.histplot(df["CGPA"], kde=True)      # no outliers present
plt.show()
```



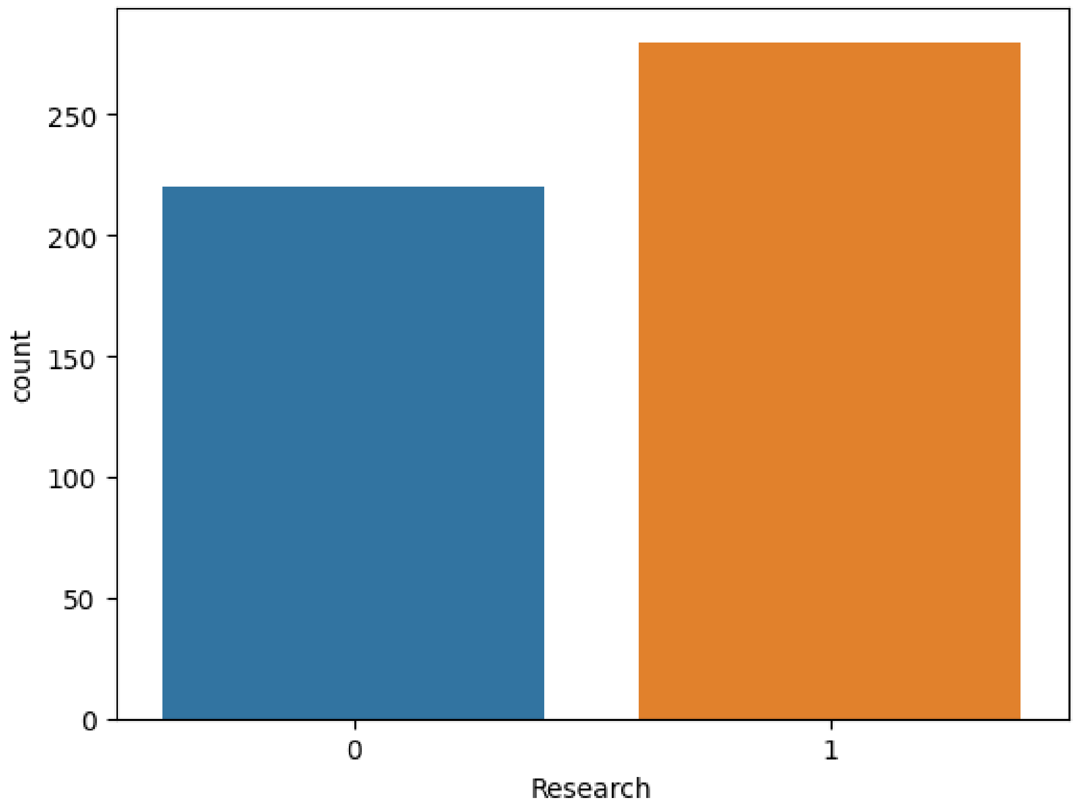
[30]:
```
df["Research"].unique()
```

[30]:
```
array([1, 0])
```

[31]:
```
df["Research"].value_counts() # Maximum applicants has Research Experience␣
    ↪score 1
```

[31]:
```
1    280
0    220
Name: Research, dtype: int64
```

[32]:
```
sns.countplot(data=df,x="Research") # Visual representation of the above code
plt.show()                          # Research experience applicants has high␣
    ↪chanse to admit
```



[33]:
```
df["Chance of Admit "].unique()
```

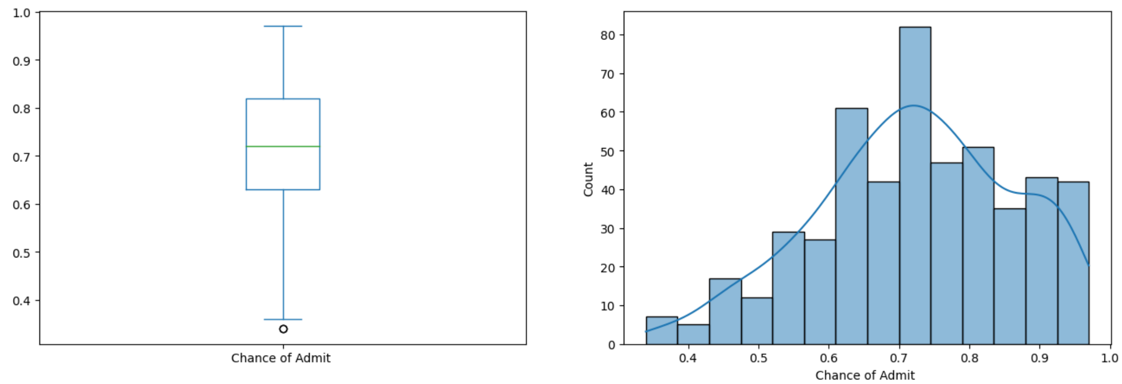
[33]:
```
array([0.92, 0.76, 0.72, 0.8 , 0.65, 0.9 , 0.75, 0.68, 0.5 , 0.45, 0.52,
       0.84, 0.78, 0.62, 0.61, 0.54, 0.66, 0.63, 0.64, 0.7 , 0.94, 0.95,
       0.97, 0.44, 0.46, 0.74, 0.91, 0.88, 0.58, 0.48, 0.49, 0.53, 0.87,
       0.86, 0.89, 0.82, 0.56, 0.36, 0.42, 0.47, 0.55, 0.57, 0.96, 0.93,
       0.38, 0.34, 0.79, 0.71, 0.69, 0.59, 0.85, 0.77, 0.81, 0.83, 0.67,
       0.73, 0.6 , 0.43, 0.51, 0.39, 0.37])
```

[34]:
```
df["Chance of Admit "].value_counts(bins=5) # Maximum applicants chances of␣
↪admit range liebetween 0.5 to 0.9
```

[34]:
```
(0.718, 0.844]    155
(0.592, 0.718]    141
(0.844, 0.97]     109
(0.466, 0.592]     71
(0.338, 0.466]     24
Name: Chance of Admit , dtype: int64
```

[35]:
```
plt.subplot(121)
df["Chance of Admit "].plot.box(figsize=(16,5))    # Median is at 0.72
```

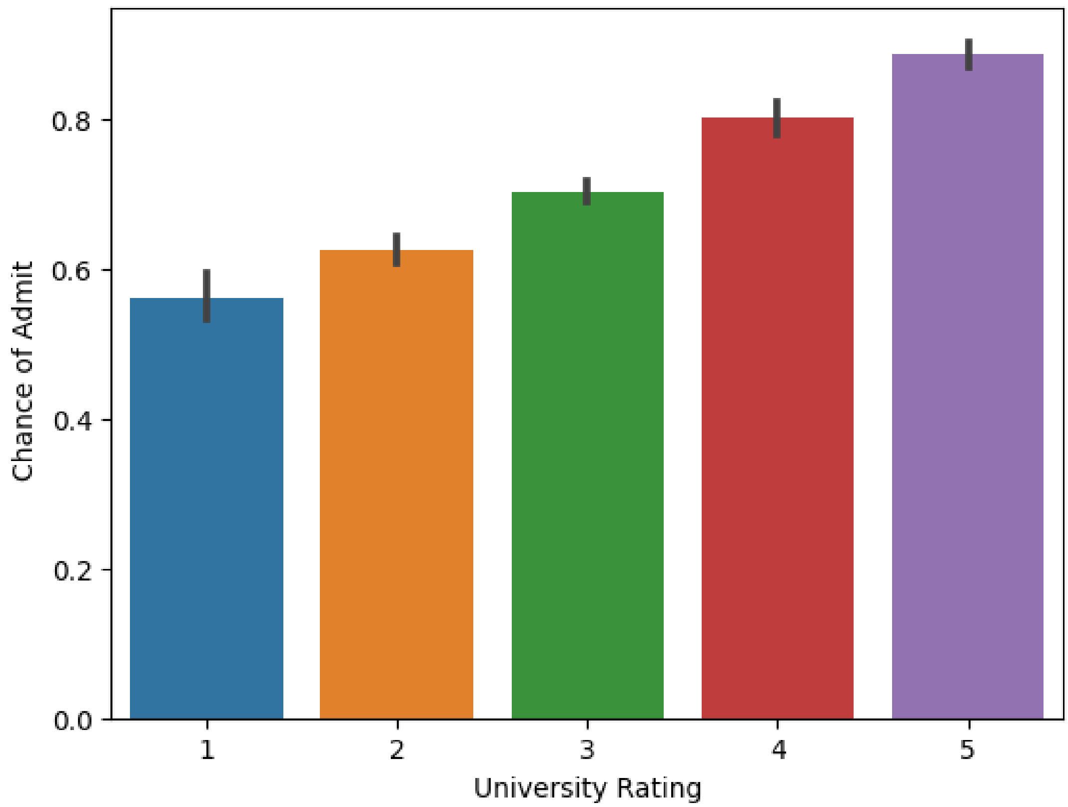
```
plt.subplot(122)                                    # Chance of admit data is
 ↪left skewed
sns.histplot(df["Chance of Admit "], kde=True)      # There are some outliers
 ↪present
plt.show()
```



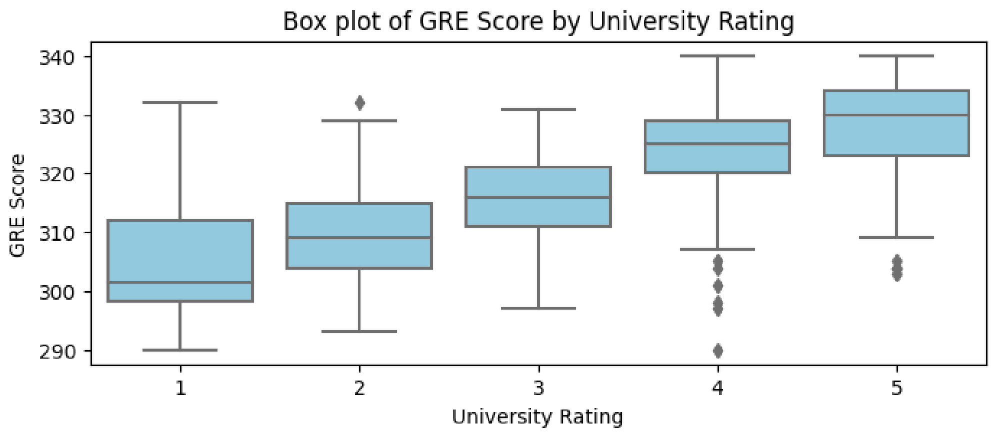
1.2 Bivariate Analysis

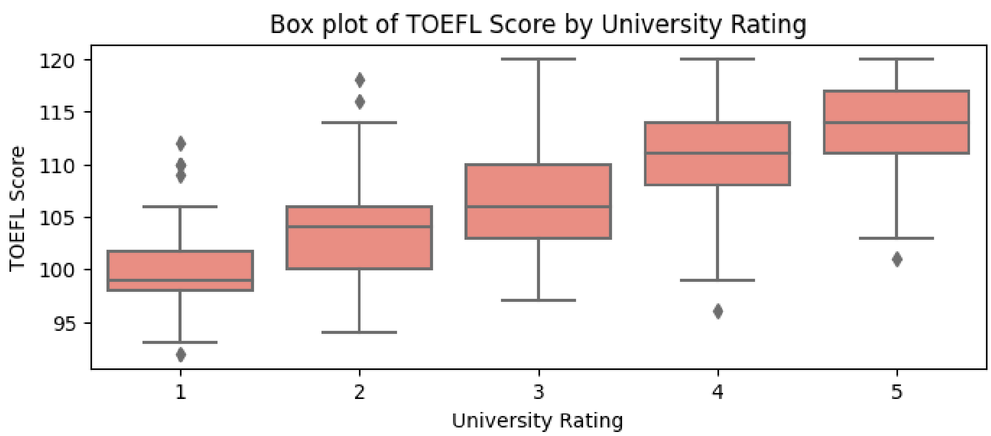
[36]:
```
sns.barplot(x="University Rating",y="Chance of Admit ",data=df,estimator=np.
 ↪mean) # University rating 3,4 and 5 has maximum chance of admit.
```

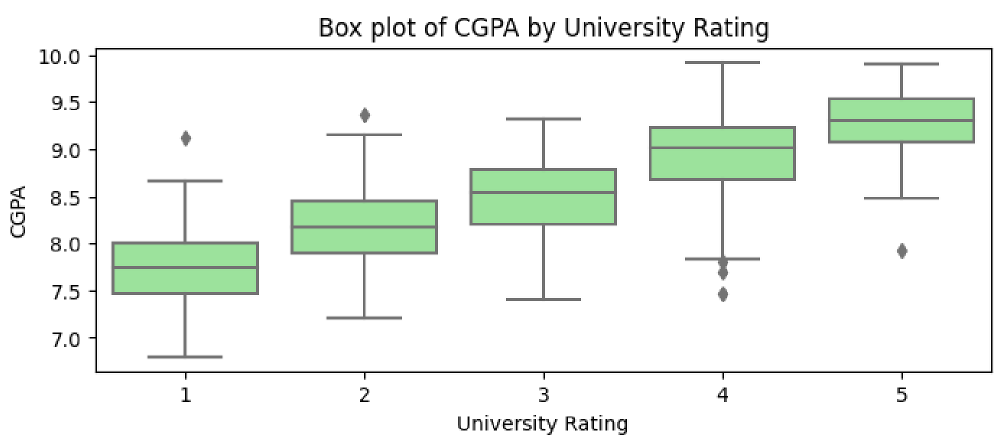
[36]: <Axes: xlabel='University Rating', ylabel='Chance of Admit '>

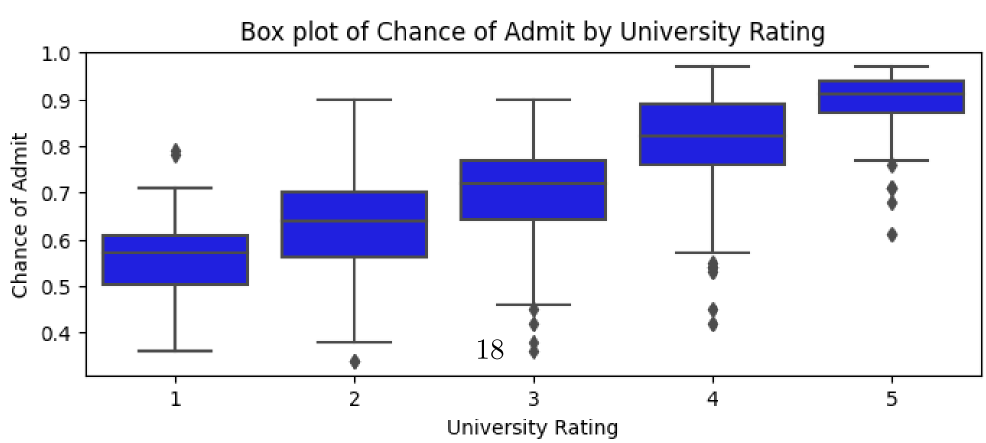


[102]:
```
# Columns to analyze
columns_to_analyze = ['GRE Score', 'TOEFL Score','CGPA','Chance of Admit']
x_column = 'University Rating'

# Set up subplots
fig, axes = plt.subplots(nrows=len(columns_to_analyze), ncols=1, figsize=(8, 4␣
↪* len(columns_to_analyze)))
fig.subplots_adjust(hspace=0.5)

# Define colors for each variable
colors = ['skyblue', 'salmon', 'lightgreen', 'blue']

for i, y_column in enumerate(columns_to_analyze):
    # Grouped box plot
    sns.boxplot(x=x_column, y=y_column, data=df, ax=axes[i],␣
↪palette=[colors[i]])
    axes[i].set_title(f'Box plot of {y_column} by {x_column}')

plt.show()
```


Box plot of GRE Score by University Rating
GRE Score
340
330
320
310
300
290
1
2
3
4
5
University Rating

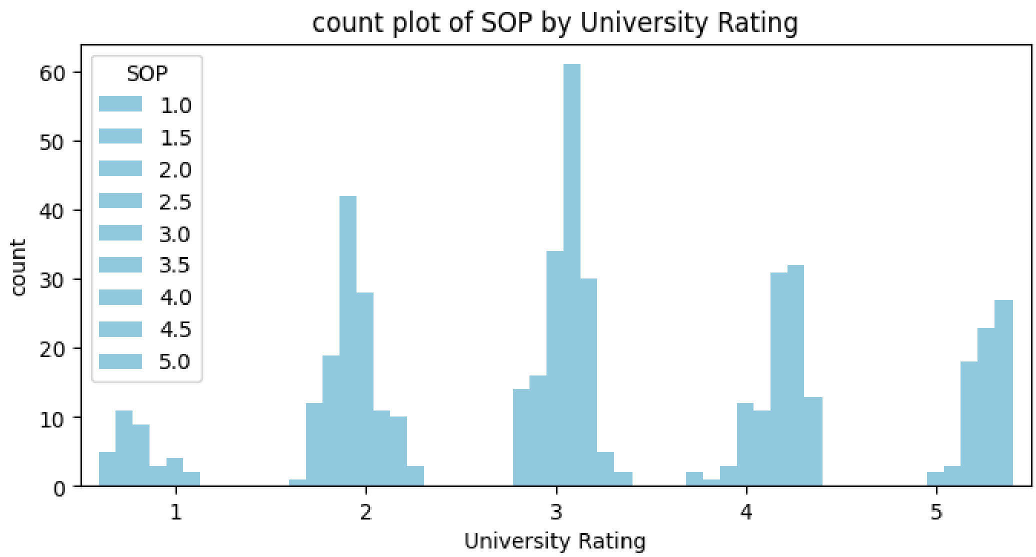

Box plot of TOEFL Score by University Rating
TOEFL Score
120
115
110
105
100
95
1
2
3
4
5
University Rating

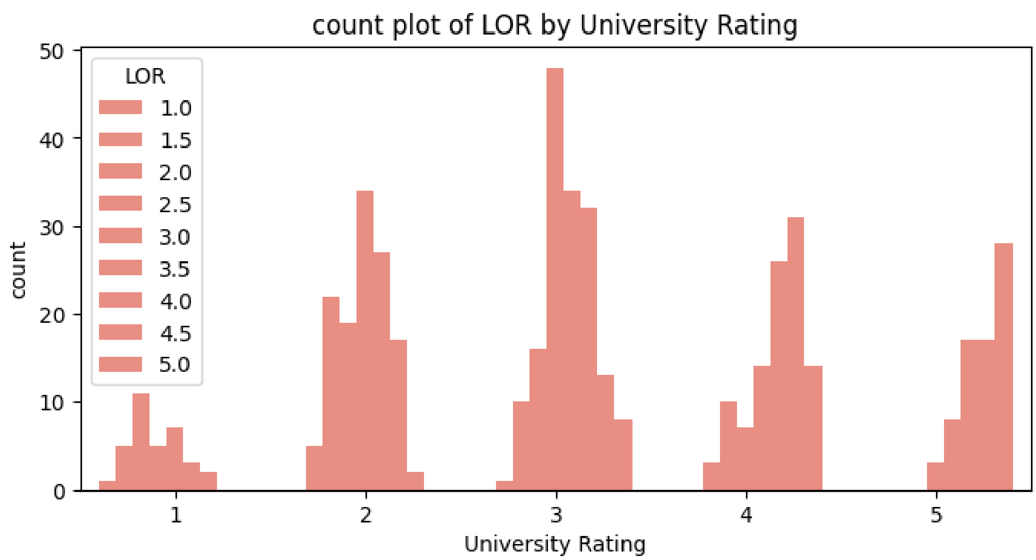

Box plot of CGPA by University Rating
CGPA
10.0
9.5
9.0
8.5
8.0
7.5
7.0
1
2
3
4
5
University Rating

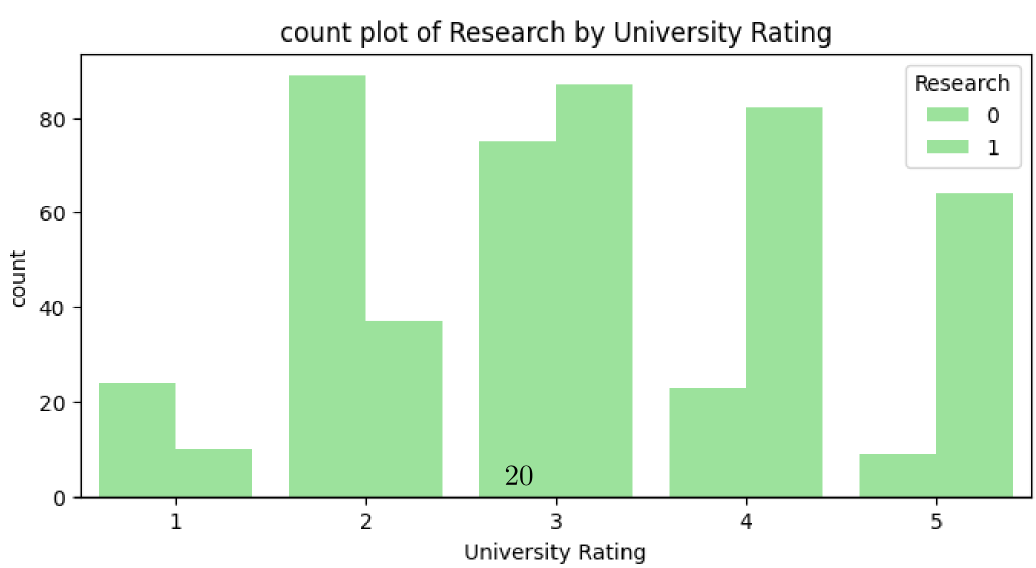

Box plot of Chance of Admit by University Rating
Chance of Admit
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1
2
3
4
5
University Rating

[109]:

```
# Columns to analyze
columns_to_analyze = ['SOP', 'LOR','Research']
x_column = 'University Rating'

# Set up subplots
fig, axes = plt.subplots(nrows=len(columns_to_analyze), ncols=1, figsize=(8, 5␣
 ↪* len(columns_to_analyze)))
fig.subplots_adjust(hspace=0.5)

# Define colors for each variable
colors = ['skyblue', 'salmon', 'lightgreen',]

for i, y_column in enumerate(columns_to_analyze):
    # Grouped box plot
    sns.countplot(x=x_column, hue=y_column, data=df, ax=axes[i],␣
 ↪palette=[colors[i]])
    axes[i].set_title(f'count plot of {y_column} by {x_column}')

plt.show()
```

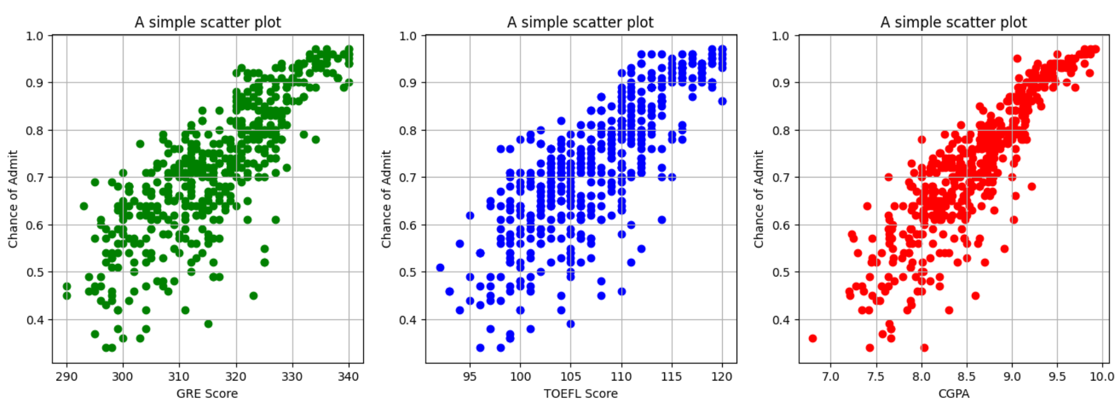

count plot of SOP by University Rating
SOP
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
count
0
10
20
30
40
50
60
1
2
3
4
5
University Rating


count plot of LOR by University Rating
LOR
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
count
0
10
20
30
40
50
1
2
3
4
5
University Rating


count plot of Research by University Rating
Research
0
1
count
0
20
40
60
80
1
2
3
4
5
University Rating

[37]:

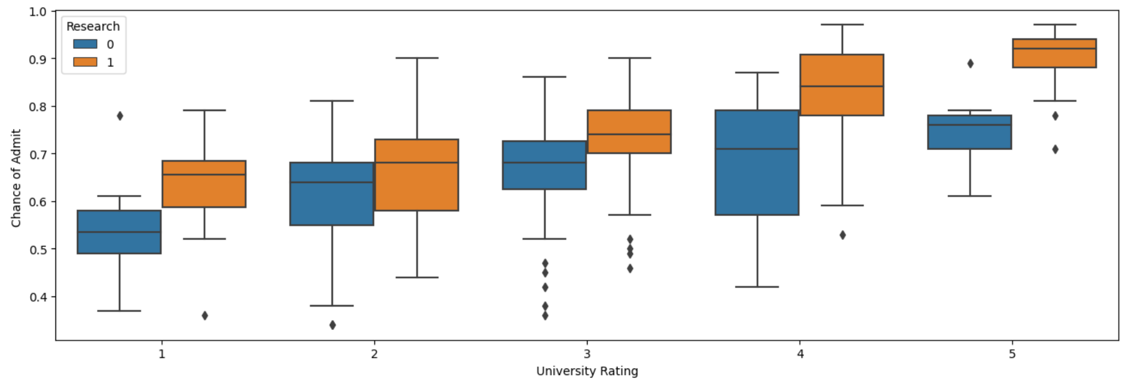
```
plt.rcParams["figure.figsize"] = (16,5)

plt.subplot(1,3,1)
plt.scatter(x="GRE Score",y="Chance of Admit ",data=df,c='g')
plt.title('A simple scatter plot')  # GRE score and chance of admit is directly proportional with each other.
plt.xlabel('GRE Score')
plt.ylabel('Chance of Admit')
plt.grid()

plt.subplot(1,3,2)
plt.scatter(x="TOEFL Score",y="Chance of Admit ",data=df,c='b')
plt.title('A simple scatter plot') # TOEFL Score and chance of admit is directly proportional with each other.
plt.xlabel('TOEFL Score')
plt.ylabel('Chance of Admit')
plt.grid()

plt.subplot(1,3,3)
plt.scatter(x="CGPA",y="Chance of Admit ",data=df,c='r')
plt.title('A simple scatter plot')  # CGPA and chance of admit is directly proportional with each other.
plt.xlabel('CGPA')
plt.ylabel('Chance of Admit')
plt.grid()
```



1.3 Mulativariate Analysis

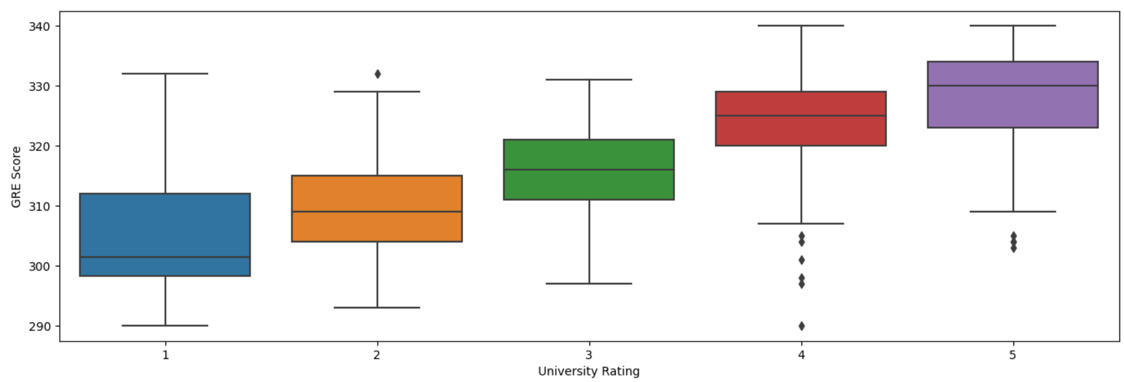
[38]:
```
sns.boxplot(x="University Rating",hue="Research",data=df,y="Chance of Admit␣
↪",dodge=True)
# applicant from university rating 4 with no research experience has more␣
↪chances of admision
```

[38]: <Axes: xlabel='University Rating', ylabel='Chance of Admit '>



[39]:
```
sns.boxplot(x="University Rating",data=df,y="GRE Score",dodge=True)
#
```

[39]: <Axes: xlabel='University Rating', ylabel='GRE Score'>



1.4 2. Data Preprocessing

1.4.1 Duplicate value check

[40]:
```
bool_series = df.duplicated() # From value count we can see that there are zero duplicate values in the data present.
bool_series.value_counts()
```

[40]:
```
False    500
dtype: int64
```

1.4.2 Missing value treatment

[41]:
```
(df.isnull().sum()/len(df))*100 # No missing value present in the data
```

[41]:
```
GRE Score            0.0
TOEFL Score          0.0
University Rating    0.0
SOP                  0.0
LOR                  0.0
CGPA                 0.0
Research             0.0
Chance of Admit      0.0
dtype: float64
```

1.4.3 Outlier treatment

[42]:
```
df.describe()
```

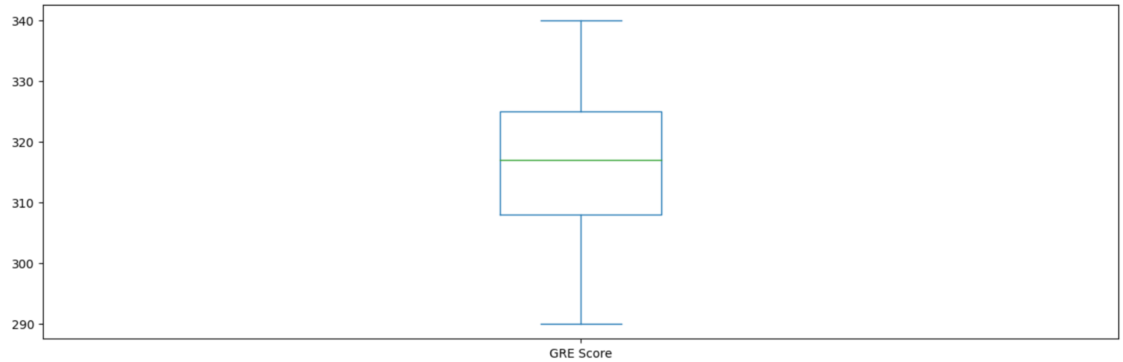
[42]:
```
        GRE Score  TOEFL Score  University Rating         SOP        LOR   \
count  500.000000   500.000000         500.000000  500.000000  500.00000
mean   316.472000   107.192000           3.114000    3.374000    3.48400
std     11.295148     6.081868           1.143512    0.991004    0.92545
min    290.000000    92.000000           1.000000    1.000000    1.00000
25%    308.000000   103.000000           2.000000    2.500000    3.00000
50%    317.000000   107.000000           3.000000    3.500000    3.50000
75%    325.000000   112.000000           4.000000    4.000000    4.00000
max    340.000000   120.000000           5.000000    5.000000    5.00000

             CGPA    Research  Chance of Admit
count  500.000000  500.000000        500.00000
mean     8.576440    0.560000          0.72174
std      0.604813    0.496884          0.14114
min      6.800000    0.000000          0.34000
25%      8.127500    0.000000          0.63000
50%      8.560000    1.000000          0.72000
75%      9.040000    1.000000          0.82000
```

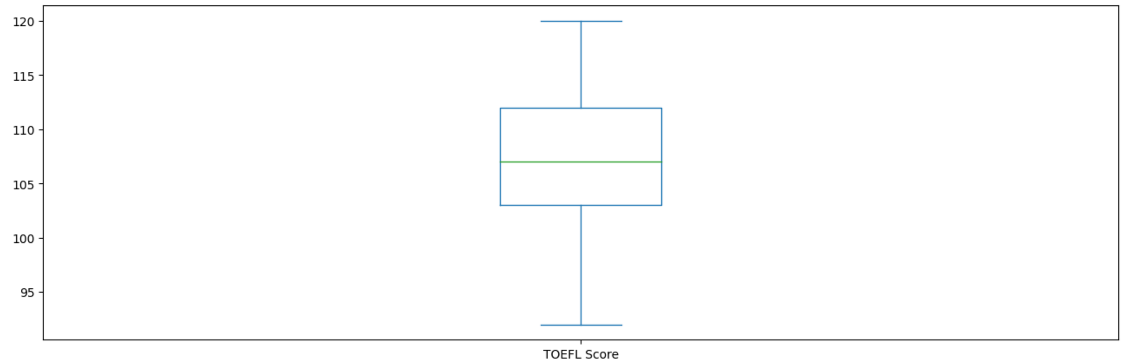
```
max      9.920000    1.000000           0.97000
```

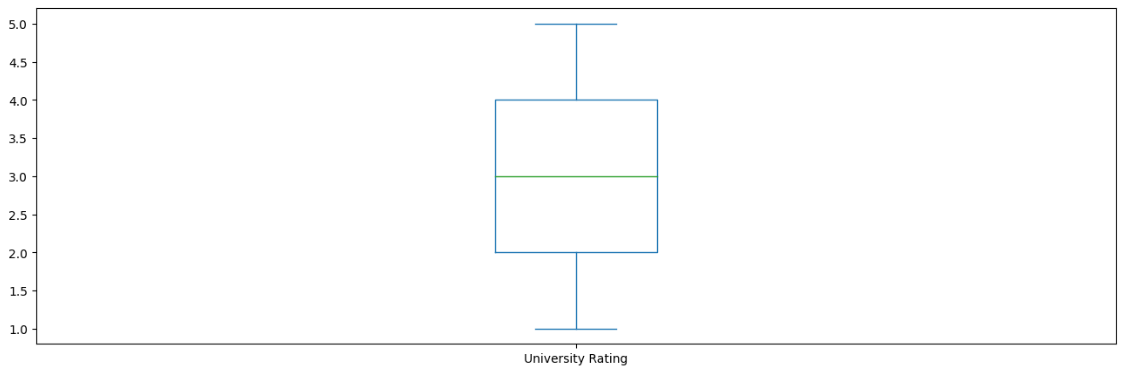
[43]:
```python
df.columns
```

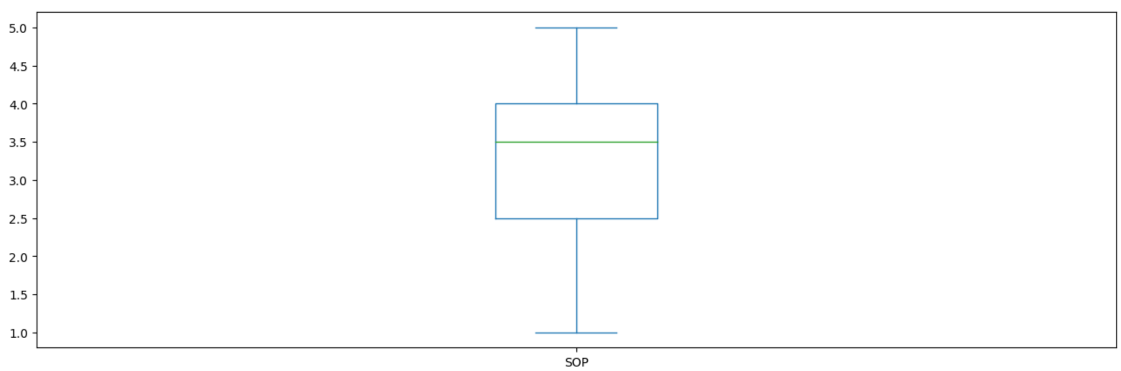
[43]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

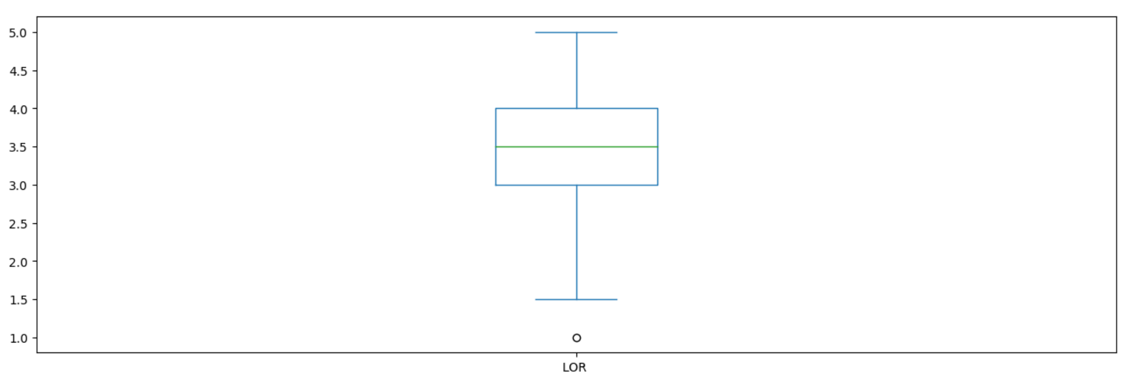
[44]:
```python
total_columns=['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ',␣
 ↪'CGPA', 'Research', 'Chance of Admit ']
for col in total_columns:
  df[col].plot.box(figsize=(16,5))
  plt.show()
```

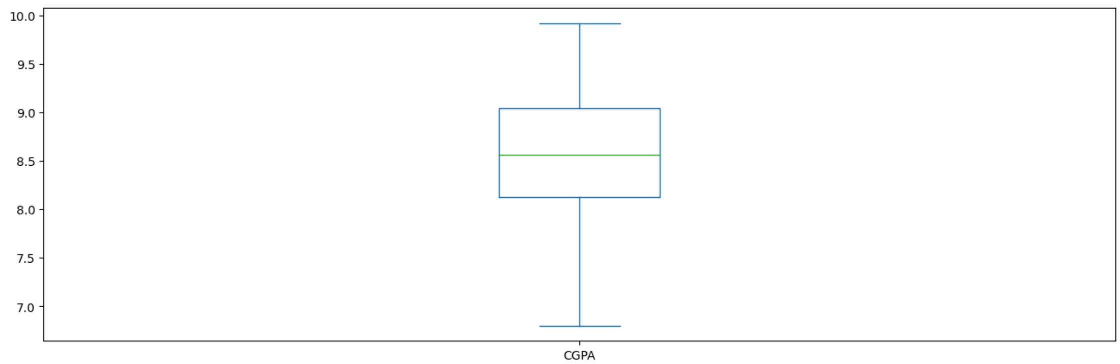


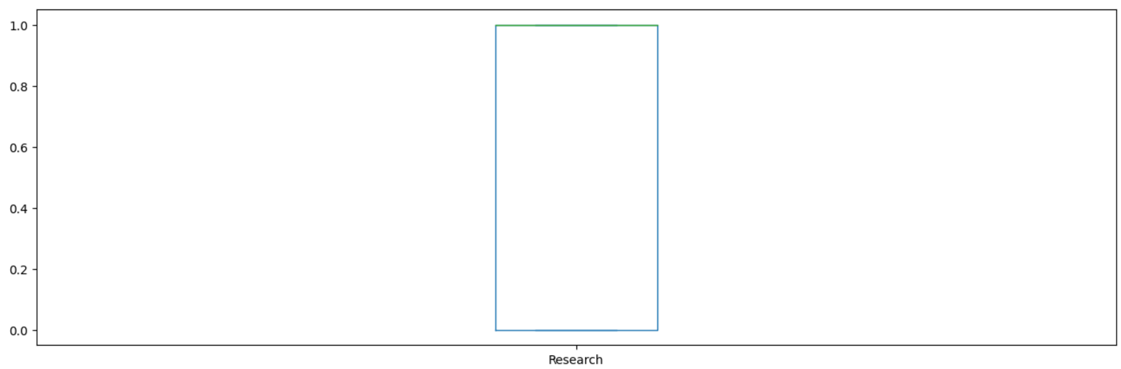


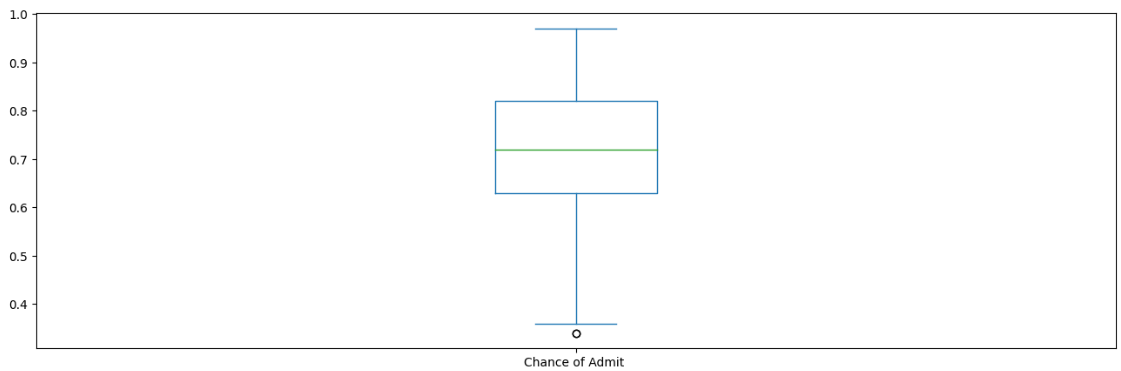

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
University Rating


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
SOP


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
LOR







[45]:

```
Q1=df['Chance of Admit '].quantile(0.25)
Q3=df['Chance of Admit '].quantile(0.75)
IQR=Q3-Q1
print(IQR)
lower_limit=Q1 - 1.5*IQR
Upper_limit=Q3 + 1.5*IQR
print(lower_limit,Upper_limit)
```

```
0.18999999999999995
0.3450000000000001 1.105
```

[46]:
```
df=df[(df['Chance of Admit ']>lower_limit) & (df['Chance of Admit␣
↪']<Upper_limit)]
```

[47]:
```
df.shape # Outliers are very less in the data so we can neglect the it.
```

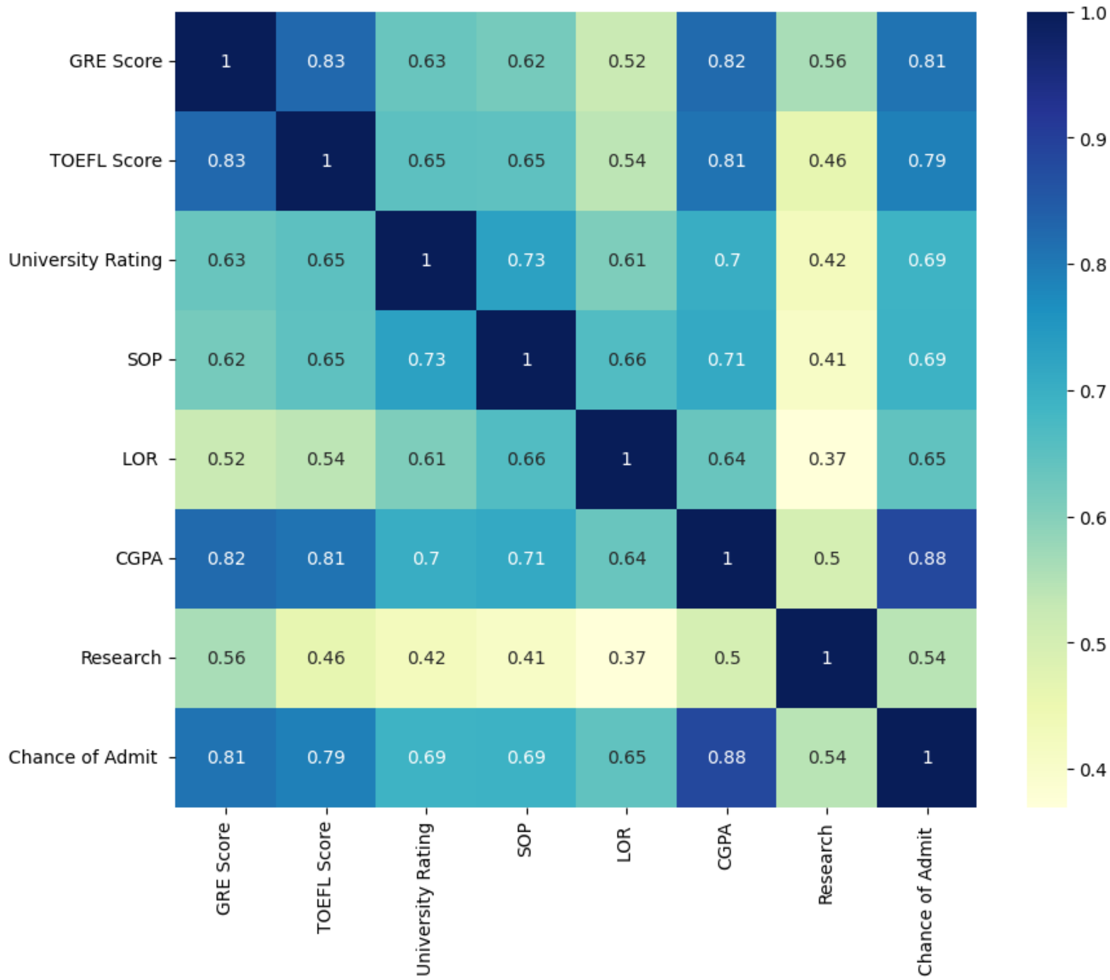
[47]:
```
(498, 8)
```

1.4.4 Correlations

[48]:
```
df.shape
```

[48]:
```
(498, 8)
```

[49]:
```
plt.figure(figsize=(10,8))
ax = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```



Some insights on correlation :

1. GRE score is highly correlated with chance of admit
2. TOEFL score is highlty correlated with chanse of admit.
3. CGPA is also highly correlated with chanse of admit.
4. University rating, SOP and LOR are almost samely correlated with taget variable which is chanse to admit.
5. Some independent variables are highly correlated with the independent variables, meaning multicollinearity is present in the data. for example GRE score is highly correlated with TOEFL score with 0.83

1.4.5 Feature engineering

[50]:
```
df=pd.read_csv("Jamboree_Admission.csv")
```

[51]:
```
# Feature engineering adding extra parameter
ratio_CGPA_GRE=(df["CGPA"]/df["GRE Score"])*100
df["ratio_CGPA_GRE"]=ratio_CGPA_GRE
```

[52]:
```
# let's combine SOP and LOR columns with name SOP_LOR_total
ratio_CGPA_TOEFL=(df["CGPA"]/df["TOEFL Score"])*100
df["ratio_CGPA_TOEFL"]=ratio_CGPA_TOEFL
```

[53]:
```
df.head()
```

[53]:
```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0           1        337          118                  4  4.5   4.5  9.65
1           2        324          107                  4  4.0   4.5  8.87
2           3        316          104                  3  3.0   3.5  8.00
3           4        322          110                  3  3.5   2.5  8.67
4           5        314          103                  2  2.0   3.0  8.21

   Research  Chance of Admit   ratio_CGPA_GRE  ratio_CGPA_TOEFL
0         1              0.92        2.863501          8.177966
1         1              0.76        2.737654          8.289720
2         1              0.72        2.531646          7.692308
3         1              0.80        2.692547          7.881818
4         0              0.65        2.614650          7.970874
```

[54]:
```
df["Chance of Admit"]=df["Chance of Admit "]
```

[55]:
```
df_new=df.drop(columns=['Chance of Admit ',"Serial No."],axis=1)
df_new.head()
```

[55]:
```
   GRE Score  TOEFL Score  University Rating  SOP  LOR  CGPA  Research  \
0        337          118                  4  4.5  4.5  9.65         1
1        324          107                  4  4.0  4.5  8.87         1
2        316          104                  3  3.0  3.5  8.00         1
3        322          110                  3  3.5  2.5  8.67         1
4        314          103                  2  2.0  3.0  8.21         0

   ratio_CGPA_GRE  ratio_CGPA_TOEFL  Chance of Admit
0        2.863501          8.177966             0.92
1        2.737654          8.289720             0.76
2        2.531646          7.692308             0.72
3        2.692547          7.881818             0.80
4        2.614650          7.970874             0.65
```

[56]:
```python
plt.figure(figsize=(10,8))
ax = sns.heatmap(df_new.corr(), cmap="YlGnBu", annot=True)
```

GRE Score, TOEFL Score and CGPA are hightest correlated with `chance of admit` in same order. - New encoded features are strong predictor.

- Still multicollinearity present in the data.

1.5 Data preparation for modeling

1.6 Standardization

[57]:
```
## scaling
## Lets scale the data, standardization
from sklearn.preprocessing import StandardScaler
```

[58]:
```
df_new.columns
```

[58]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'ratio_CGPA_GRE', 'ratio_CGPA_TOEFL', 'Chance of Admit'],
      dtype='object')
```

[59]:
```
df_num=df_new[['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ',
 ↪'CGPA','Research', 'ratio_CGPA_GRE', 'ratio_CGPA_TOEFL']]
```

[60]:
```
scaler = StandardScaler()
df_sc=scaler.fit_transform(df_num)
```

[61]:
```
df_new_sc=pd.DataFrame(df_sc, columns=df_num.columns, index=df_num.index)
df_new_sc.head()
```

[61]:
```
   GRE Score  TOEFL Score  University Rating       SOP       LOR       CGPA  \
0   1.819238     1.778865           0.775582  1.137360  1.098944  1.776806
1   0.667148    -0.031601           0.775582  0.632315  1.098944  0.485859
2  -0.041830    -0.525364          -0.099793 -0.377773  0.017306 -0.954043
3   0.489904     0.462163          -0.099793  0.127271 -1.064332  0.154847
4  -0.219074    -0.689952          -0.975168 -1.387862 -0.523513 -0.606480

   Research  ratio_CGPA_GRE  ratio_CGPA_TOEFL
0  0.886405        1.257447          0.529523
1  0.886405        0.240787          0.863755
2  0.886405       -1.423461         -0.922986
3  0.886405       -0.123617         -0.356197
4 -1.128152       -0.752910         -0.089850
```

[62]:
```
df_new1=pd.concat([df_new_sc,df_new["Chance of Admit"]],axis=1)
```

[63]:
```
df_new1.head() # dataframe ready for the modeling
```

[63]:
```
   GRE Score  TOEFL Score  University Rating       SOP       LOR       CGPA  \
0   1.819238     1.778865           0.775582  1.137360  1.098944  1.776806
```

```
1   0.667148    -0.031601           0.775582  0.632315  1.098944  0.485859
2  -0.041830    -0.525364          -0.099793 -0.377773  0.017306 -0.954043
3   0.489904     0.462163          -0.099793  0.127271 -1.064332  0.154847
4  -0.219074    -0.689952          -0.975168 -1.387862 -0.523513 -0.606480

   Research  ratio_CGPA_GRE  ratio_CGPA_TOEFL  Chance of Admit
0  0.886405        1.257447          0.529523             0.92
1  0.886405        0.240787          0.863755             0.76
2  0.886405       -1.423461         -0.922986             0.72
3  0.886405       -0.123617         -0.356197             0.80
4 -1.128152       -0.752910         -0.089850             0.65
```

[64]:
```python
df_new1.shape
```

[64]:
```
(500, 10)
```

1.7 Model building

1.7.1 Simple linear regression

[65]:
```python
x = df_new1["CGPA"].values # CGPA is 0.88 correlated with taget variable i.e.␣
↪chanse of admit.
y = df_new1["Chance of Admit"].values
```

[66]:
```python
def hypothesis(x,weights):
  y_hat=weights[0]+ weights[1]*x
  return y_hat
```

[67]:
```python
hypothesis(2.3,[5,0.8]) ## randomly predicted value
```

[67]:
```
6.84
```

[68]:
```python
def error(x,y,weights):
  n= len(x)
  err=0
  for i in range(n):
    y_hat_i=hypothesis(x[i],weights)
    err=err+(y[i] - y_hat_i)**2
  return err/n
```

[69]:
```python
def gradient(x,y,weights):
    n=len(x)
    grade= np.zeros((2, ))
    for i in range(n):
        y_hat_i=hypothesis(x[i],weights)
        grade[0] += (y_hat_i - y[i])
        grade[1] += (y_hat_i - y[i])*x[i]
```

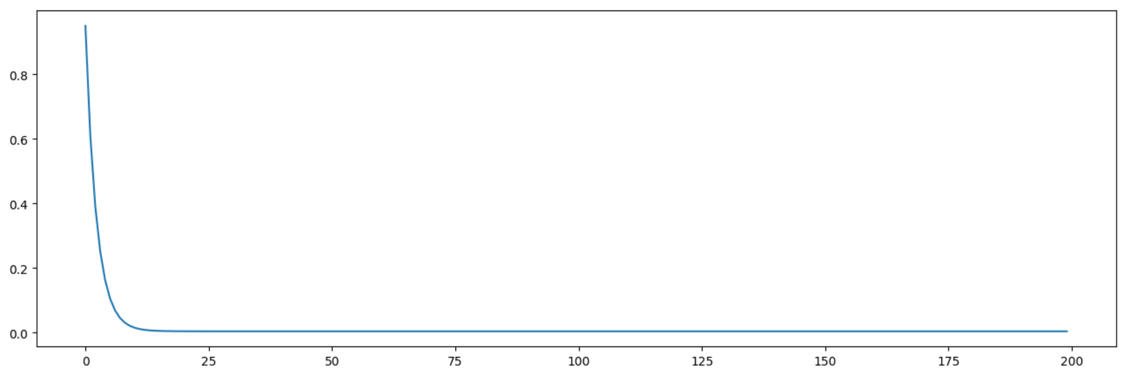
```
    return (2*grade)/n
```

[70]:
```
def gradient_descent(x,y,ran_itr=200,learning_rate=0.1):
    '''step1: initialise the variable '''
    weights=np.random.rand(2)
    ''' step2: rpeate for 100 times'''
    error_list=[]
    for i in range(ran_itr):
        e=error(x,y,weights)
        error_list.append(e)
        grade = gradient(x,y,weights)
        weights[0]=weights[0]-learning_rate*grade[0]
        weights[1]=weights[1]-learning_rate*grade[1]

    return weights.round(3),error_list
```

[71]:
```
opt_weights, error_list=gradient_descent(x,y)
```

[72]:
```
plt.plot(error_list)
```

[72]:
```
[<matplotlib.lines.Line2D at 0x7ba27cc3a2c0>]
```



[73]:
```
Y_hat=hypothesis(x,opt_weights)
```

[74]:
```
def r2_score(Y, Y_hat):
    num = np.sum((Y - Y_hat)**2)
    denom = np.sum((Y - Y.mean())**2)

    r2 = 1 - num/denom

    return r2.round(3)
```

[75]:
```
r2_score(y,Y_hat) # performance of the simple linear regression model using␣
↪CGPA veriable is 78%
                  # Only CGPA is not important to check the chanse od admit␣
↪hence let's check multivarient linear regression
```

[75]: 0.779

2 Building the Linear Regression model and commenting on the model statistics and model coefficients with column names

[76]:
```
df_new1.head()
```

[76]:
```
   GRE Score  TOEFL Score  University Rating       SOP       LOR      CGPA  \
0   1.819238     1.778865           0.775582  1.137360  1.098944  1.776806
1   0.667148    -0.031601           0.775582  0.632315  1.098944  0.485859
2  -0.041830    -0.525364          -0.099793 -0.377773  0.017306 -0.954043
3   0.489904     0.462163          -0.099793  0.127271 -1.064332  0.154847
4  -0.219074    -0.689952          -0.975168 -1.387862 -0.523513 -0.606480

   Research  ratio_CGPA_GRE  ratio_CGPA_TOEFL  Chance of Admit
0  0.886405        1.257447          0.529523             0.92
1  0.886405        0.240787          0.863755             0.76
2  0.886405       -1.423461         -0.922986             0.72
3  0.886405       -0.123617         -0.356197             0.80
4 -1.128152       -0.752910         -0.089850             0.65
```

[77]:
```
# Statmodels implementation of Linear regression
import statsmodels.api as sm


X = df_new1[df_new1.columns.drop('Chance of Admit')]
Y = df_new1["Chance of Admit"]


X_sm = sm.add_constant(X)  #Statmodels default is without intercept, to add␣
↪intercept we need to add constant

sm_model = sm.OLS(Y, X_sm).fit()

print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Chance of Admit   R-squared:                       0.823
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     252.5
```

```
Date:                Mon, 04 Dec 2023   Prob (F-statistic):          1.02e-177
Time:                        16:29:48   Log-Likelihood:                 702.37
No. Observations:                 500   AIC:                            -1385.
Df Residuals:                     490   BIC:                            -1343.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const                 0.7217      0.003    269.021      0.000       0.716
0.727
GRE Score             0.1270      0.079      1.607      0.109      -0.028
0.282
TOEFL Score          -0.0343      0.089     -0.386      0.700      -0.209
0.140
University Rating     0.0067      0.004      1.538      0.125      -0.002
0.015
SOP                   0.0014      0.005      0.316      0.752      -0.007
0.010
LOR                   0.0156      0.004      4.066      0.000       0.008
0.023
CGPA                 -0.0739      0.121     -0.611      0.542      -0.312
0.164
Research              0.0123      0.003      3.736      0.000       0.006
0.019
ratio_CGPA_GRE        0.1357      0.101      1.345      0.179      -0.062
0.334
ratio_CGPA_TOEFL     -0.0377      0.065     -0.583      0.560      -0.165
0.089
==============================================================================
Omnibus:                      118.043   Durbin-Watson:                   0.804
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              283.371
Skew:                          -1.198   Prob(JB):                     2.93e-62
Kurtosis:                       5.803   Cond. No.                         148.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

3 Linear Regression model

[78]:
```
X = df_new1[df_new1.columns.drop('Chance of Admit')]
Y = df_new1["Chance of Admit"]

#Train and test data split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=100)

from sklearn.linear_model import LinearRegression
lr = LinearRegression()

# train the model
lr.fit(X_train, y_train)
Pred = lr.predict(X_test)
from sklearn.metrics import r2_score,mean_squared_error, mean_absolute_error
print("Linear Regression R2_score :",r2_score(y_test, Pred))
```

```
Linear Regression R2_score : 0.8313554590045338
```

[79]:
```
lr.coef_
```

[79]:
```
array([ 0.07362709,  0.03097081,  0.00572688, -0.00115692,  0.01779132,
       -0.05144605,  0.01351941,  0.07205956,  0.00848471])
```

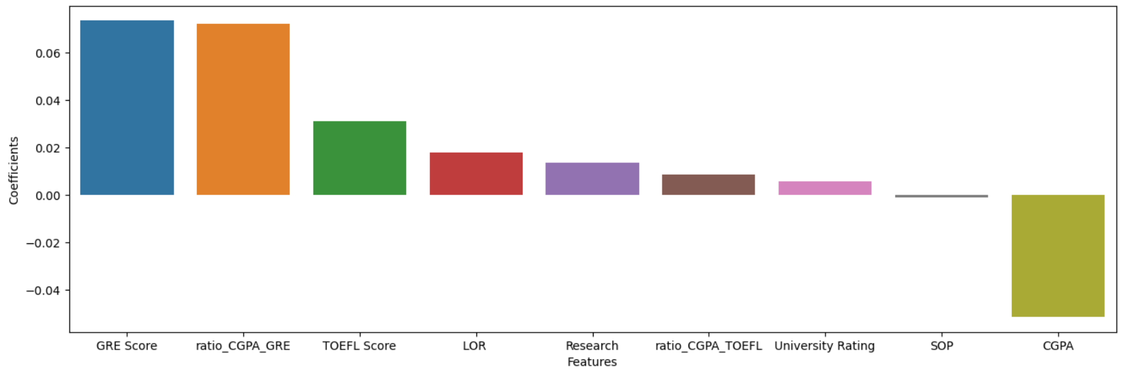
[80]:
```
coeff=pd.DataFrame()                                    # GRE score has highest␣
↪weight than the other features
X_c=X                                                   # 3rd highest weight is␣
↪on CGPA score.
coeff["Features"]=X_c.columns
coeff["Coefficients"]=lr.coef_
coeff["Coefficients"] = round(coeff["Coefficients"], 5)
coeff = coeff.sort_values(by = "Coefficients", ascending = False)
coeff
```

[80]:

	Features	Coefficients
0	GRE Score	0.07363
7	ratio_CGPA_GRE	0.07206
1	TOEFL Score	0.03097
4	LOR	0.01779
6	Research	0.01352
8	ratio_CGPA_TOEFL	0.00848
2	University Rating	0.00573
3	SOP	-0.00116
5	CGPA	-0.05145

[81]:
```
sns.barplot(x="Features",y="Coefficients",data=coeff) # visual represntation of␣
↪the cofficients of all features present in the data.
```

[81]: <Axes: xlabel='Features', ylabel='Coefficients'>



The bar graph shows the coefficients of all features present in the data. The features are listed on the x-axis, and their corresponding coefficients are on the y-axis. The coefficient values range from -0.04 to 0.06.

The features with the highest coefficients are GRE Score, TOEFL Score, and CGPA. This means that these features are the most predictive of the target variable. The features with the lowest coefficients are University Rating and SOP. This means that these features are the least predictive of the target variable.

The coefficient for Research is positive, which means that a higher research score is associated with a higher target variable. The coefficient for LOR is negative, which means that a higher number of letters of recommendation is associated with a lower target variable.

Overall, the data analysis suggests that GRE Score, TOEFL Score, and CGPA are the most important factors for predicting the target variable. Research is also a positive predictor, while LOR is a negative predictor. University Rating and SOP are the least important factors.

4 Lasso regression using sklearn

[82]:
```
from sklearn.linear_model import Lasso

X = df_new1[df_new1.columns.drop('Chance of Admit')]
Y = df_new1["Chance of Admit"]

#Train and test data split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=100)
```

```
# initialize Lasso regression and set the value of alpha equal to 1
ls = Lasso(alpha= 1)

# fit the model
ls.fit(X_train,y_train)

#predict
ls_pred=ls.predict(X_test)
#r2_score
lasso_r2_score=r2_score(y_test, ls_pred)

#print intercepts and coefficients rounded off upto 2 decimal digit
print("Coefficients:",list(zip(X.columns, ls.coef_)))
print("Intercepts:",ls.intercept_.round(2))
print("LASSO R2_score:",lasso_r2_score)
```

```
Coefficients: [('GRE Score', 0.0), ('TOEFL Score', 0.0), ('University Rating',
0.0), ('SOP', 0.0), ('LOR ', 0.0), ('CGPA', 0.0), ('Research', 0.0),
('ratio_CGPA_GRE', 0.0), ('ratio_CGPA_TOEFL', 0.0)]
Intercepts: 0.72
LASSO R2_score: -0.0424956830527512
```

Note:- Here, in this data set all feature are important there is no as such less important feature hence we can not make all the features equal to zero as it has some multicolinearity but we can not remove it by lasso regression. Hence we can canclude that lasso regression is not suitable for this dataset.

5 Ridge regression using sklearn

[83]:
```
from sklearn.linear_model import Ridge

rd=Ridge()
rd.fit(X_train, y_train)

#predict
rd_pred=ls.predict(X_test)
#r2_score
ridge_r2_score=r2_score(y_test, rd_pred)

#print intercepts and coefficients rounded off upto 2 decimal digit
print("Coefficients:",list(zip(X.columns, rd.coef_)))
print("Intercepts:",rd.intercept_.round(2))
print("Ridge R2_score:",ridge_r2_score.round(5))
```

```
Coefficients: [('GRE Score', 0.03614361074336035), ('TOEFL Score',
0.032593241490360254), ('University Rating', 0.005798732819659048), ('SOP',
```

```
-0.0009687211589251483), ('LOR ', 0.0177264699326378), ('CGPA',
0.020028152564601543), ('Research', 0.013413003702513708), ('ratio_CGPA_GRE',
0.024366648155271887), ('ratio_CGPA_TOEFL', 0.009882680902879417)]
Intercepts: 0.72
Ridge R2_score: -0.0425
```

Note:- Same with the ridge regression there is no need to regularise the model as each feature has it's own importance and without making it zero or moving it toward zero we can build the linear regression model with zero mean_square_error value and r2 score upto 0.8+

6 Testing the assumptions of the linear regression model

6.0.1 1.Multicollinearity check by VIF score (variables are dropped one-by-one till none has VIF>5)

[84]:
```
# VIF (Variance Inflation Factor)
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[85]:
```
vif = pd.DataFrame()
X_t = X
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.
 ↪shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

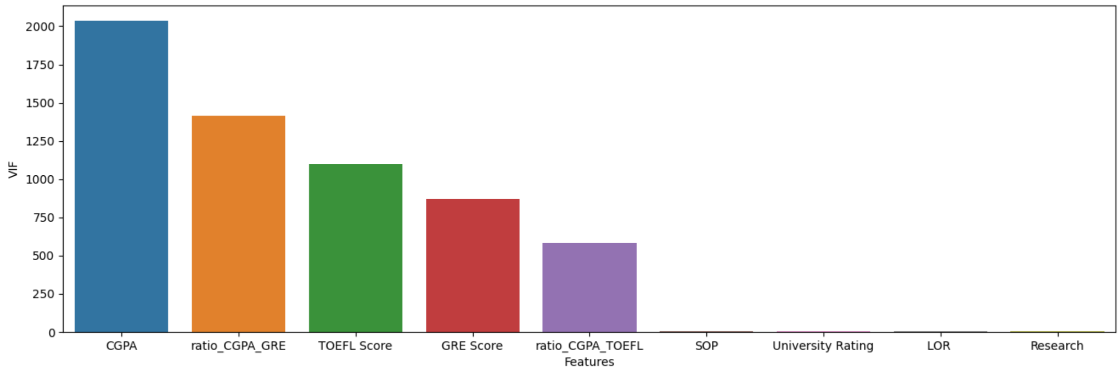
[85]:

	Features	VIF
5	CGPA	2036.53
7	ratio_CGPA_GRE	1413.88
1	TOEFL Score	1095.53
0	GRE Score	867.55
8	ratio_CGPA_TOEFL	580.79
3	SOP	2.84
2	University Rating	2.67
4	LOR	2.04
6	Research	1.50

[86]:
```
sns.barplot(x="Features",y="VIF",data=vif) # visual rpresntation of VIF for␣
 ↪each feature.
                                            # Any variable with a VIF of 10 or␣
 ↪above is considered strongly correlated with other variables.
                                            # CGPA, TOEFL score, GRE score these␣
 ↪all original feature are highly correlated with other variables.
                                            # SOP, University rating, LOR and␣
 ↪Reseach's VIF is less than 5 hence they are not correlated with other␣
 ↪variables.
```

```
                                    # new encoded feature ratio_CGPA_GRE␣
↪and ratio_CGPA_TOEFL are highly correlated with other variables.
```

[86]: <Axes: xlabel='Features', ylabel='VIF'>



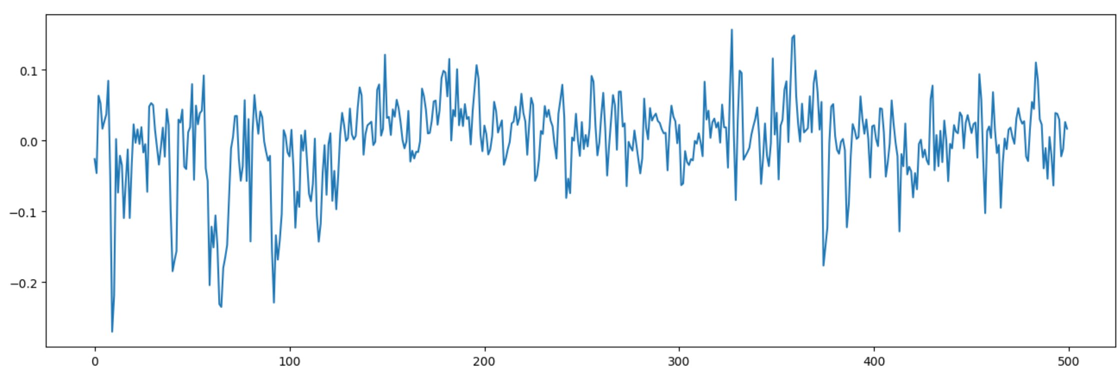
CGPA has the highest importance score, followed by ratio_CGPA_GRE, TOEFL Score, and GRE Score. This suggests that a student's CGPA is the most important factor in predicting their VIP score, followed by how their CGPA compares to their GRE score, their TOEFL score, and their GRE score itself.

The importance scores of ratio CGPA TOEFL, SOP, University Rating, LOR, and Research are all relatively low. This suggests that these factors are not as important as the others in predicting a student's VIP score.

6.1 2. The mean of residuals is nearly zero

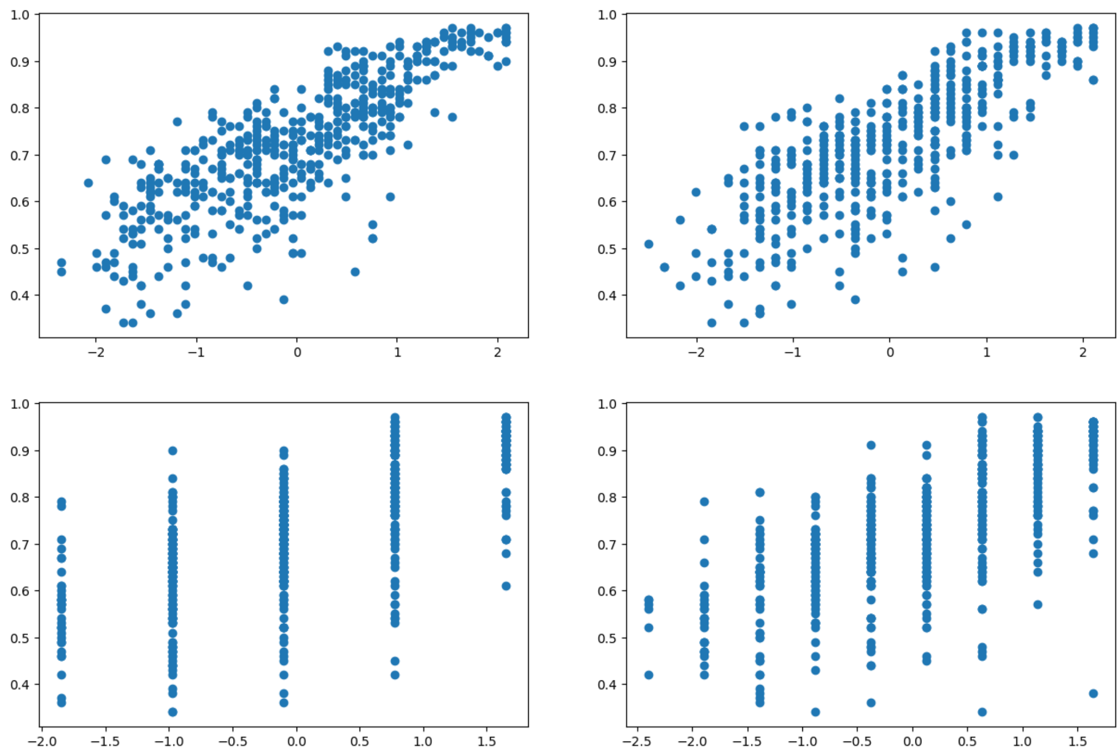
[87]:
```
residuals=sm_model.resid
plt.plot(residuals.index,residuals)
```

[87]: [<matplotlib.lines.Line2D at 0x7ba276adcc40>]



6.2 3. Linearity of variables (no pattern in the residual plot)

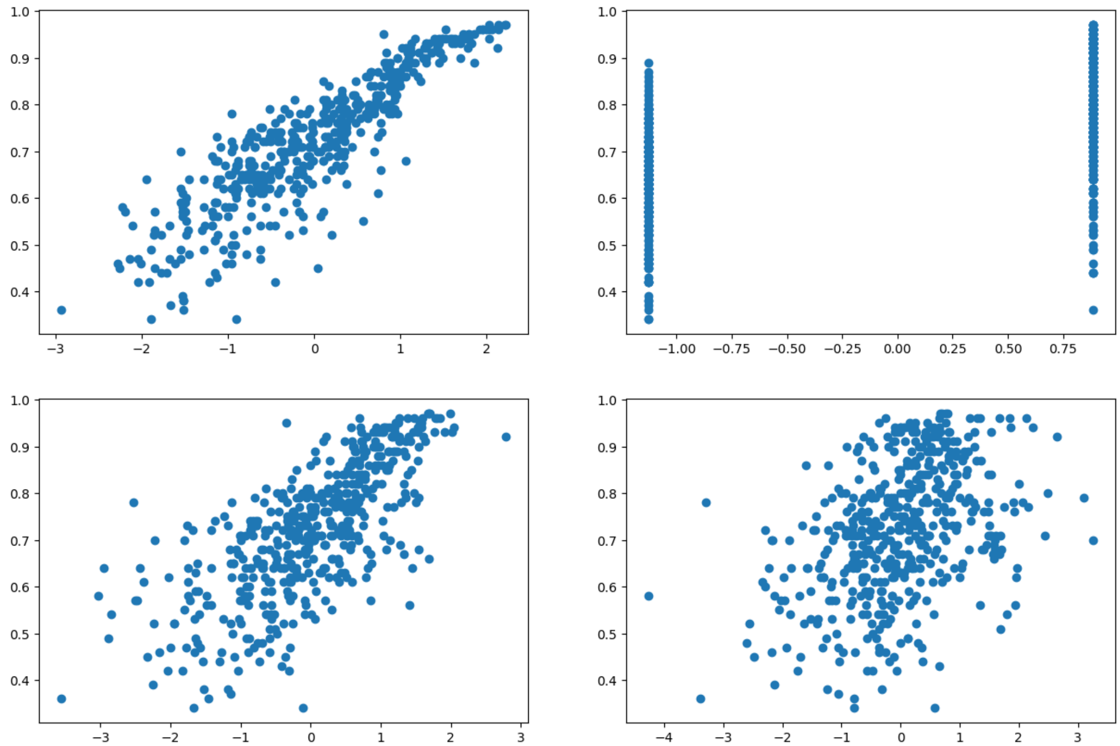
[88]:
```
from mpl_toolkits.mplot3d import axes3d                         # In this dataset␣
↪almost each variable is linearly related with the target variable.
plt.rcParams["figure.figsize"]=(15,10)                          # for example GRE␣
↪scoire is linearly related with tha chance to admit
fig,((ax1,ax2),(ax3,ax4))=plt.subplots(2,2)
ax1.scatter(X["GRE Score"],Y)
ax2.scatter(X["TOEFL Score"],Y)
ax3.scatter(X["University Rating"],Y)
ax4.scatter(X["SOP"],Y)
plt.show()
```



[88]:
```

```

[89]:
```
from mpl_toolkits.mplot3d import axes3d                    # CGPA is linearly related␣
↪with tha chance to admit column.
plt.rcParams["figure.figsize"]=(15,10)
fig,((ax1,ax2),(ax3,ax4))=plt.subplots(2,2)
ax1.scatter(X["CGPA"],Y)
ax2.scatter(X["Research"],Y)
ax3.scatter(X["ratio_CGPA_GRE"],Y)
ax4.scatter(X["ratio_CGPA_TOEFL"],Y)
```

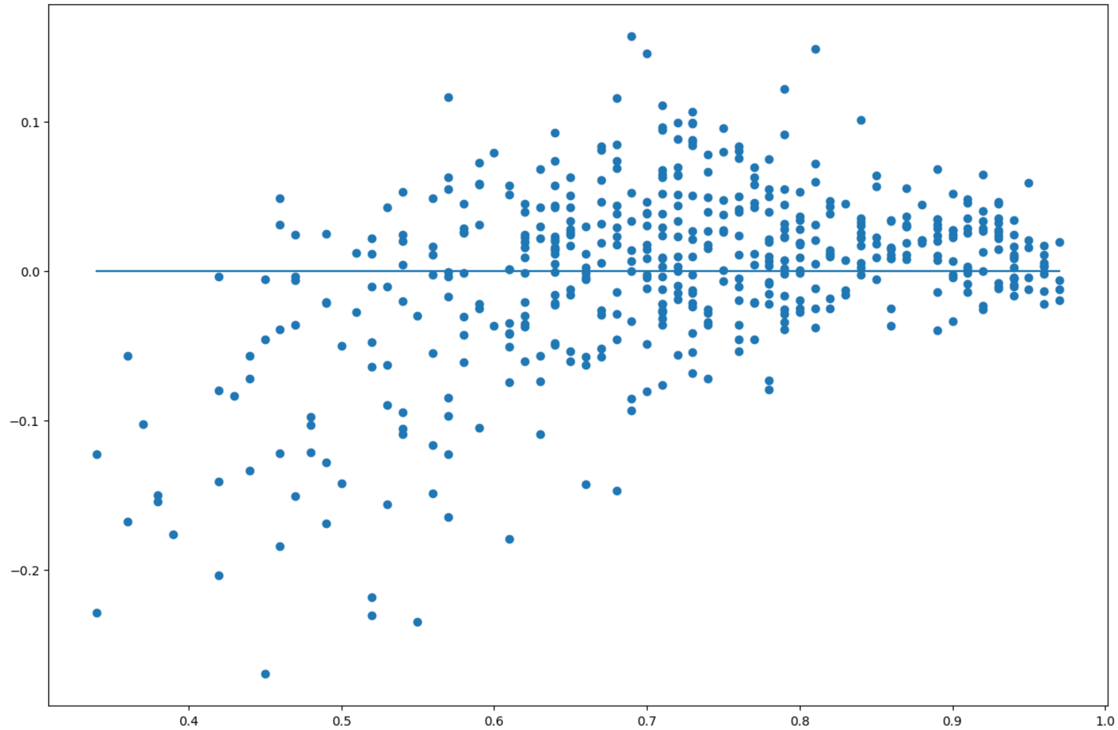
```
plt.show()
```



6.3 4. Test for Homoscedasticity

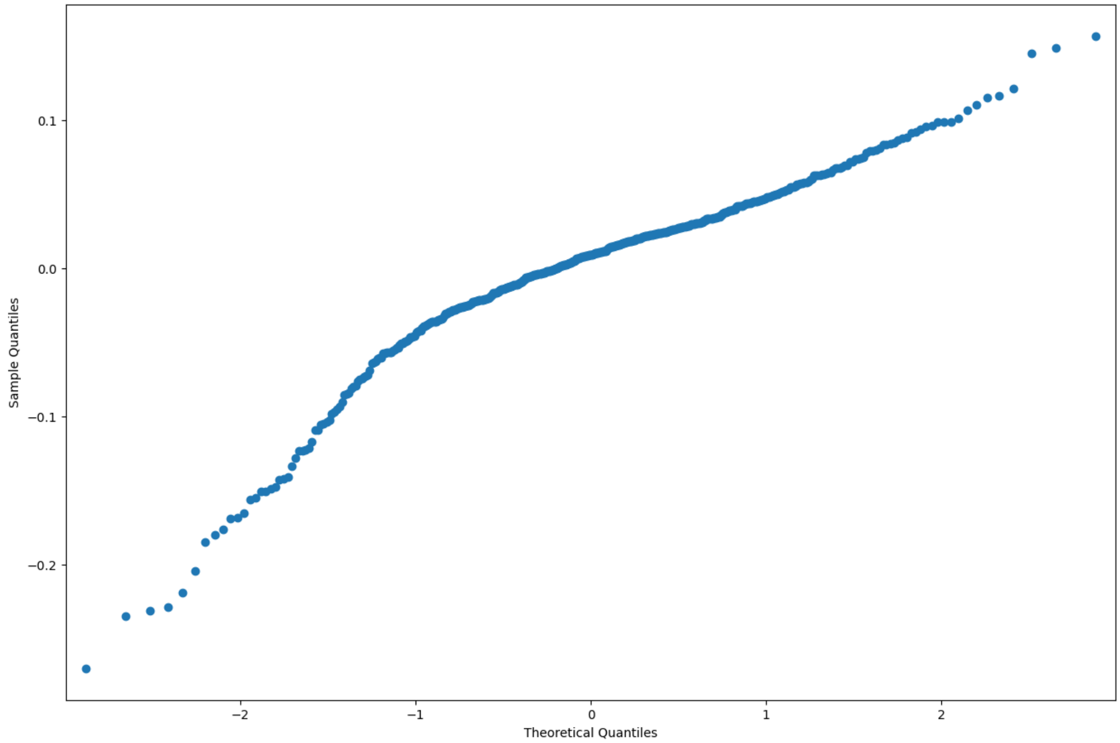
[90]:
```
residuals=sm_model.resid   # In prob/stats proof of linear regression, we␣
 ↪assume the error are normally distributed with mean of 0 and contant␣
 ↪standard deviation.
plt.scatter(Y,residuals)   # Homoscedasticity exists in our data.
plt.plot(Y,[0]*len(Y))     # There is no outliers present in the dataset.
```

[90]:
```
[<matplotlib.lines.Line2D at 0x7ba2767b2ce0>]
```



6.4 5. Normality of residuals (almost bell-shaped curve in residuals distribution, points in QQ plot are almost all on the line)

[91]:
```
residuals=sm_model.resid      # from qqplot we can say that normality of␣
 ↪residuals is linear in nature.
sm.qqplot(residuals)
plt.show()
```

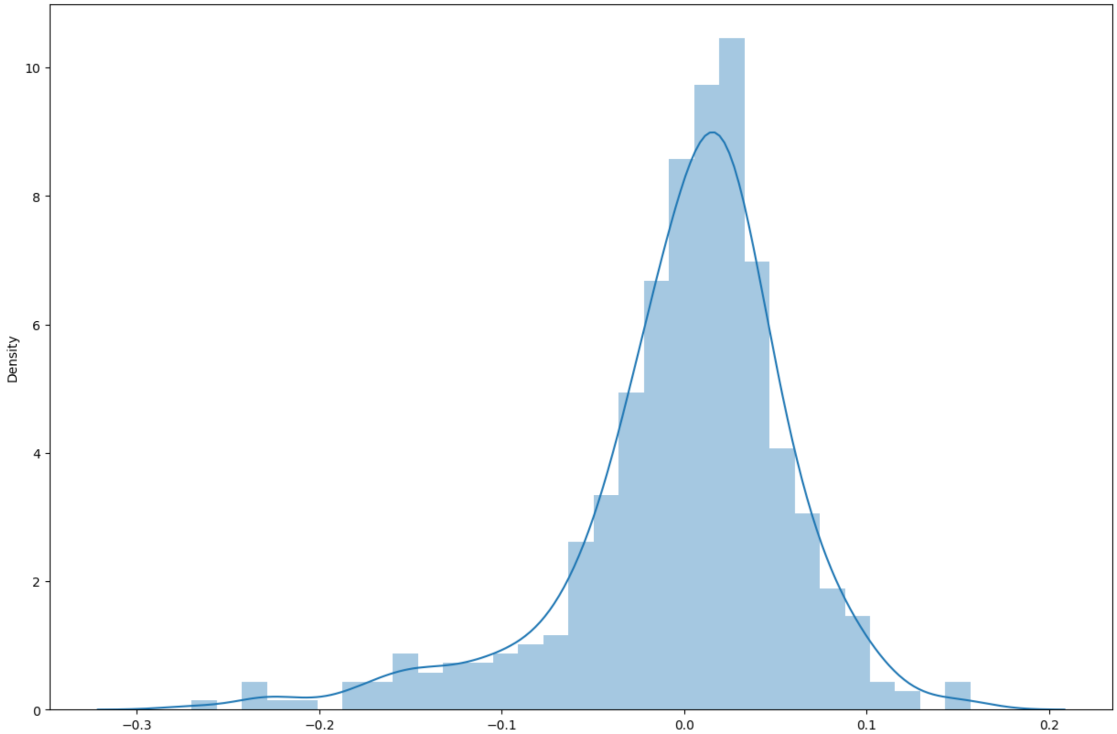


[92]:
```
np.mean(residuals)
```

[92]: 1.0746958878371515e-16

[93]:
```
sns.distplot(residuals) # Perfectly normaly distributed cure we found here.
                        # we get almost bell-shaped curve in residuals␣
 ↪distribution.
                        # There is very less outliers present in the dataset
```

[93]: <Axes: ylabel='Density'>



6.5 Model performance evaluation

6.6 Metrics checked - MAE, RMSE, R2, Adj R2

[94]:
```
from sklearn.metrics import r2_score,mean_squared_error, mean_absolute_error
```

[95]:
```
predict=lr.predict(X_test)
```

[96]:
```
MSE=mean_squared_error(y_test, predict)
print("Mean_absolute_error=",mean_absolute_error(y_test, predict).round(3))
print("Root_mean_squared_error=",np.sqrt(MSE).round(3))
print("R2_score=",r2_score(y_test, predict).round(3))
```

```
Mean_absolute_error= 0.044
Root_mean_squared_error= 0.057
R2_score= 0.831
```

[97]:
```
#Adjusted R2 score
Adj_r2_score=1 - (1-lr.score(X, Y))*(len(Y)-1)/(len(Y)-X.shape[1]-1)
print("Adjusted R2 score=",np.round(Adj_r2_score,3))
```

```
Adjusted R2 score= 0.818
```

6.7 * Mean_absolute_error(MAE) is 0.044

6.8 * Root_mean_squared_error(RMSE) is 0.057

6.9 * R2_score(R2) is 0.831

6.10 * Adjusted R2 score(Adj R2) is 0.818

[97]:

```

```

6.11 Train and test performances are checked

[98]:

```python
predict_train=lr.predict(X_train)
predict_test=lr.predict(X_test)

print("r2_score of train data=",r2_score(y_train, predict_train).round(3))
print("r2_score of test data=",r2_score(y_test, predict_test).round(3))
print()
print("mean_squared_error of train data=",mean_squared_error(y_train,␣
↪predict_train).round(3))
print("mean_squared_error of test data=",mean_squared_error(y_test,␣
↪predict_test).round(3))
print()
print("mean_absolute_error of train data=",mean_absolute_error(y_train,␣
↪predict_train).round(3))
print("mean_absolute_error of test data=",mean_absolute_error(y_test,␣
↪predict_test).round(3))
```

```
r2_score of train data= 0.818
r2_score of test data= 0.831

mean_squared_error of train data= 0.004
mean_squared_error of test data= 0.003

mean_absolute_error of train data= 0.043
mean_absolute_error of test data= 0.044
```

Comments on the performance measures

- R2 score of train data and test data is almost same there is only the difference of 0.013
- A value of 0.8 for R-square score sounds good. It means linear regression model is performing pretty good.
- Mean square error and mean absolute error is almost zero it means that model is pefectly build.
- linear regression model is performing very well on the unseen data which is test data.

[98]:

```

```

7 Actionable Insights & Recommendations:-

1. CGPA and Research are the only two variables which are important in making the prediction for Chance of Admit.

2. CGPA is the most important varibale in making the prediction for the Chance of Admit.

3. Following are the final model results on the test data:

- Mean_absolute_error(MAE) is 0.044
- Root_mean_squared_error(RMSE) is 0.057
- R2_score(R2) is 0.831
- Adjusted R2 score(Adj R2) is 0.818

4. The linear regression model or a feature where students/learners can come to their website and check if their probability of getting into the IVY league college has built and this model gives 81% true result or we can say the probability of getting admition.**

5. This model is useful to attract a maximum number of audience or students/learners and jamboree will get the basic information about that audience for the marketing purpose.**

6. With the help of this model, Jamboree can get the list of student/learner who has less chance to admit and Jamboree can offer them coaching and help them to get into their dream universities. This point is very useful from a business perspective.**

7. One recommendation while collecting the data we can create one more column of city or region names so that we can get the target audience from those particular regions and marketing can be done according to that region.**

8. This model could identify students who have lower probabilities of admission. This information could be used by Jamboree to offer coaching and support services to help improve these students' chances of admission. This approach can be beneficial both for the students and for Jamboree's business goals.

[98]:
```

```