In [56]:
```
import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
import os
```

In [57]:
```
path="C:\\Users\\moham\\Documents\\JNTU_DataScience\\JNTU_machine_learning\\ml_assignments\\M

df=pd.read_csv(path)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

In [58]:
```
df.head()
```

Out[58]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	down
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	down
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	up

In [59]:

```
sex={'Male':1,'Female':2}
df['sex']=df['sex'].replace(sex)

dataset={'Cleveland':1,'Hungary':2,'Switzerland':3,'VA Long Beach':4}
df['dataset']=df['dataset'].replace(dataset)

cp={'typical angina':1,'asymptomatic':2,'non-anginal':3,'atypical angina':4}
df['cp']=df['cp'].replace(cp)

fbs={'False':0,'True':1}
df['fbs']=df['fbs'].replace(fbs)

restecg={'lv hypertrophy':1,'normal':2,'st-t abnormality':3}
df['restecg']=df['restecg'].replace(restecg)

exang={'False':0,'True':1}
df['exang']=df['exang'].replace(exang)

slope={'downsloping':1,'flat':2,'upsloping':3}
df['slope']=df['slope'].replace(slope)

thal={'fixed defect':1,'normal':2,'reversable defect':3}
df['thal']=df['thal'].replace(thal)
```

In [60]:

```
#df.head()
#df['dataset'].values
#df.info()
df.describe()
```

Out[60]:

	id	age	sex	dataset	cp	trestbps	chol	restecg	th
count	920.000000	920.000000	920.000000	920.000000	920.000000	861.000000	890.000000	918.000000	865.000
mean	460.500000	53.510870	1.210870	2.238043	2.550000	132.132404	199.130337	1.990196	137.545
std	265.725422	9.424685	0.408148	1.130673	0.852379	19.066070	110.780810	0.632552	25.926
min	1.000000	28.000000	1.000000	1.000000	1.000000	0.000000	0.000000	1.000000	60.000
25%	230.750000	47.000000	1.000000	1.000000	2.000000	120.000000	175.000000	2.000000	120.000
50%	460.500000	54.000000	1.000000	2.000000	2.000000	130.000000	223.000000	2.000000	140.000
75%	690.250000	60.000000	1.000000	3.000000	3.000000	140.000000	268.000000	2.000000	157.000
max	920.000000	77.000000	2.000000	4.000000	4.000000	200.000000	603.000000	3.000000	202.000

In [61]:

```
df.isna().sum()
```

Out[61]:
```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
ca          611
thal        486
num           0
dtype: int64
```

In [62]:
```
df.isna().mean()
```

Out[62]:
```
id          0.000000
age         0.000000
sex         0.000000
dataset     0.000000
cp          0.000000
trestbps    0.064130
chol        0.032609
fbs         0.097826
restecg     0.002174
thalch      0.059783
exang       0.059783
oldpeak     0.067391
slope       0.335870
ca          0.664130
thal        0.528261
num         0.000000
dtype: float64
```

In [63]:
```
df.isna().sum()
```

Out[63]:
```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
ca          611
thal        486
num           0
dtype: int64
```

In [64]:
```
df.fillna(df.isna().mean(), inplace=True)
```

In [65]:
```
df.isna().sum()
```

Out[65]:
```
id          0
age         0
sex         0
dataset     0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalch      0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
num         0
dtype: int64
```

In [66]:
```
df['num'].unique()
```

Out[66]:
```
array([0, 2, 1, 3, 4], dtype=int64)
```

In [67]:
```
len(df['num'])
```

Out[67]:
```
920
```

In [68]:
```
df['num']
```

Out[68]:
```
0      0
1      2
2      1
3      0
4      0
      ..
915    1
916    0
917    2
918    0
919    1
Name: num, Length: 920, dtype: int64
```

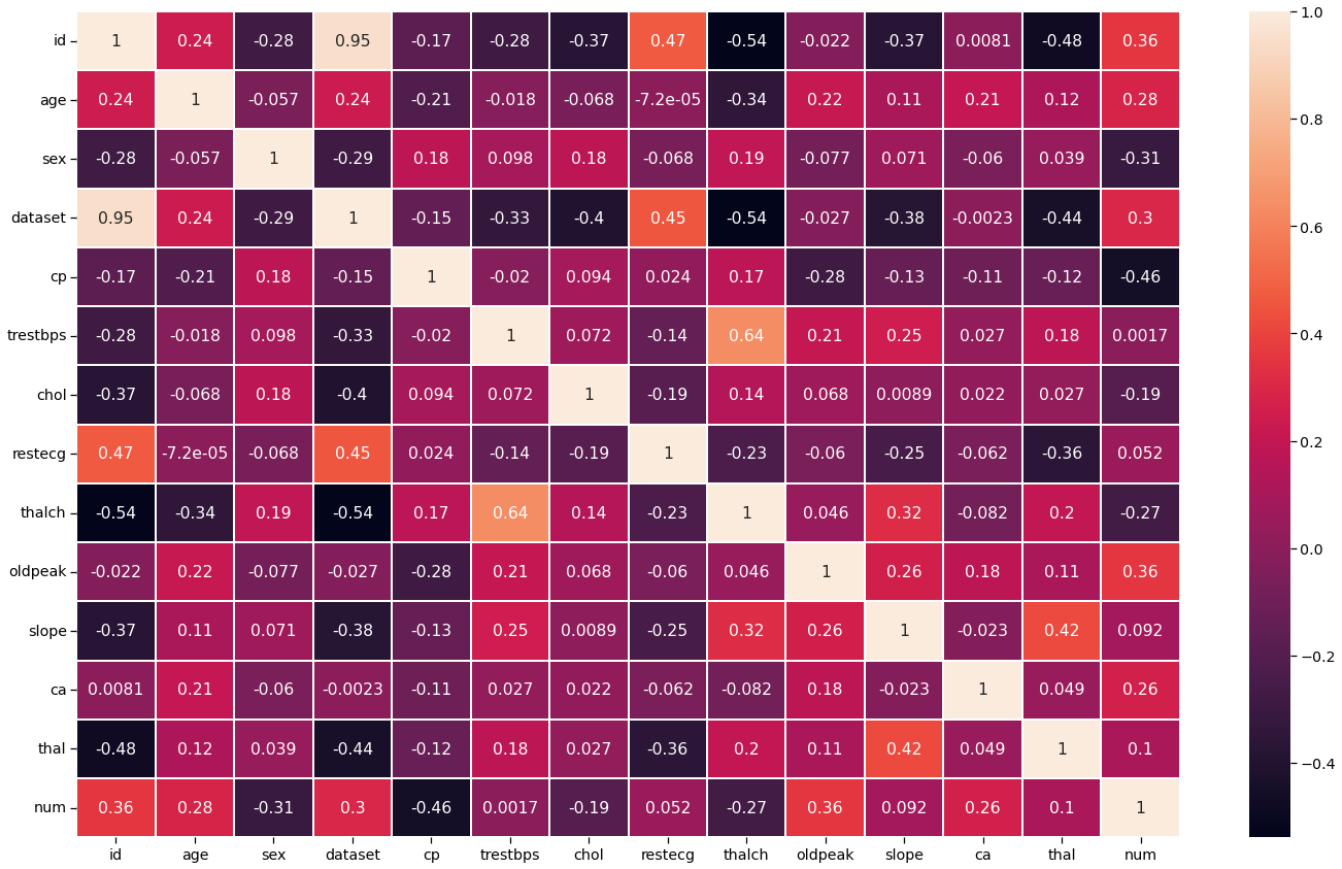
In [69]:
```
#a=0
#b=1

for i in range(0,920):
    if df['num'].values[i]>=1:
        df['num'].values[i]=1
    else:
        df['num'].values[i]=0
```

In [70]:
```
df['num'].values
```

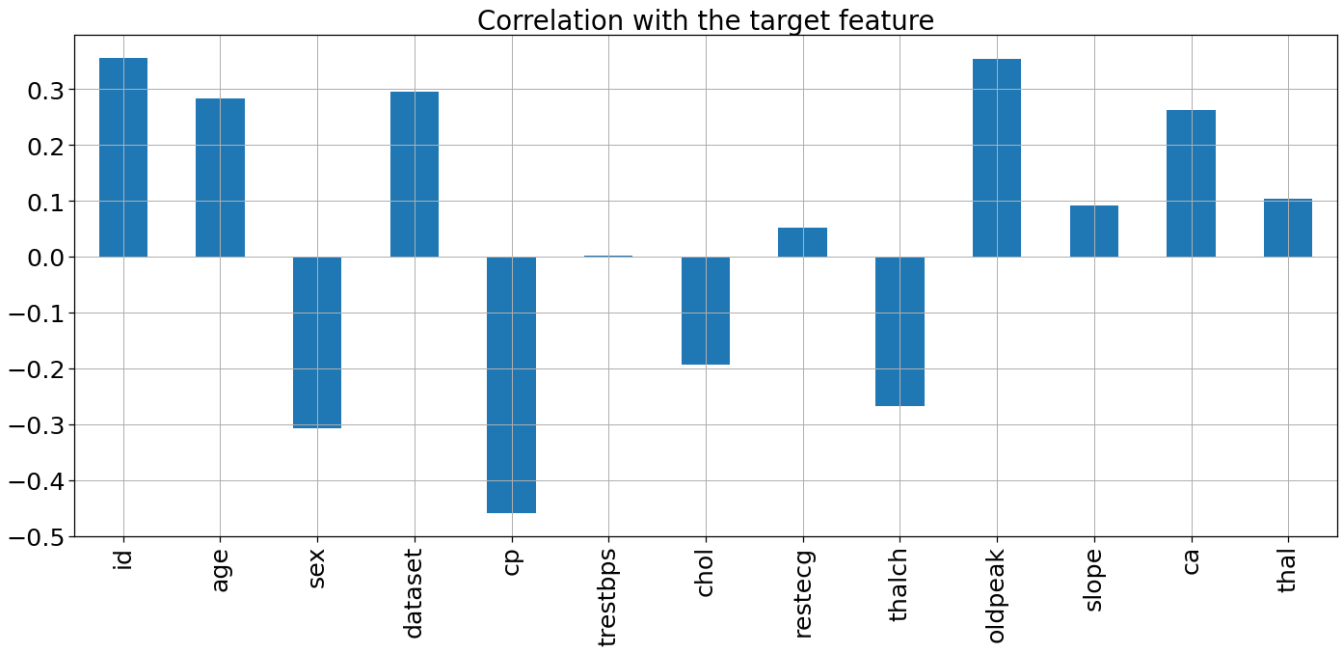
Out[70]:
```
array([0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1], dtype=int64)
```

In [71]:
```
pyplot.figure(figsize=(20,12))
sns.set_context('notebook',font_scale = 1.3)
sns.heatmap(df.corr(),annot=True,linewidth =2)
pyplot.tight_layout()
```



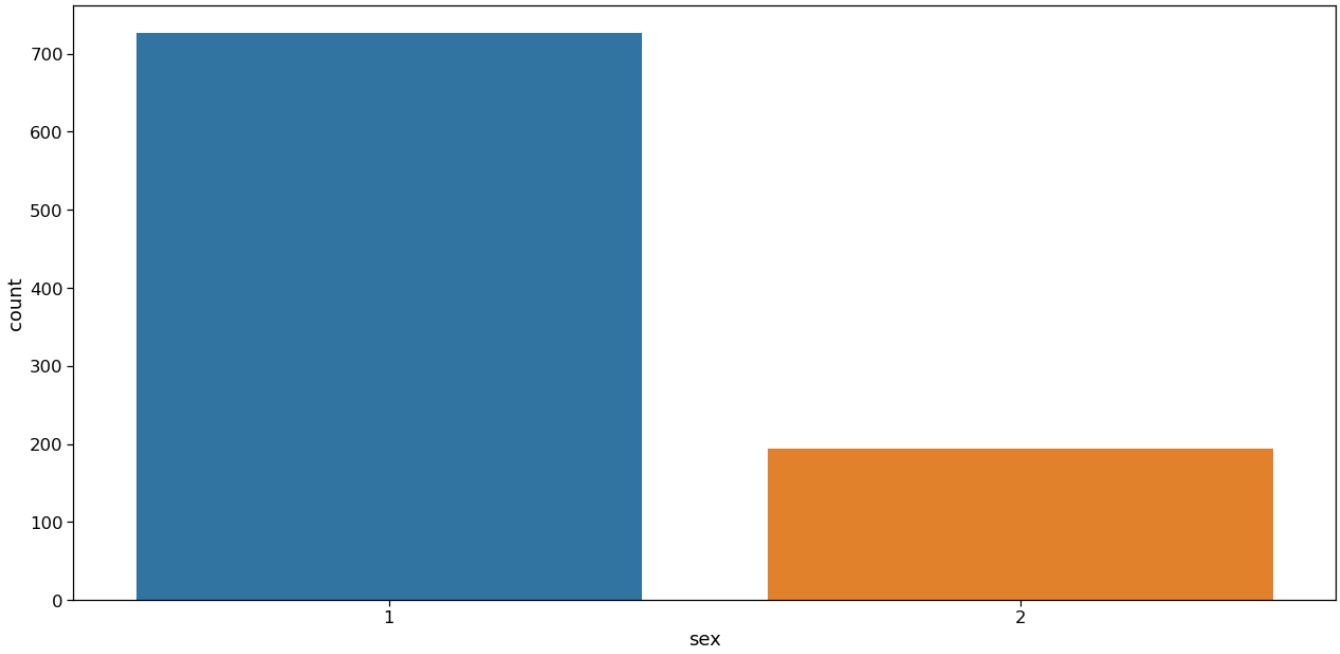
sex analysis

In [72]:
```
sns.set_context('notebook',font_scale = 2.3)
df.drop('num', axis=1).corrwith(df.num).plot(kind='bar', grid=True, figsize=(20, 10),
                                                    title="Correlation with the target fe
pyplot.tight_layout()
```



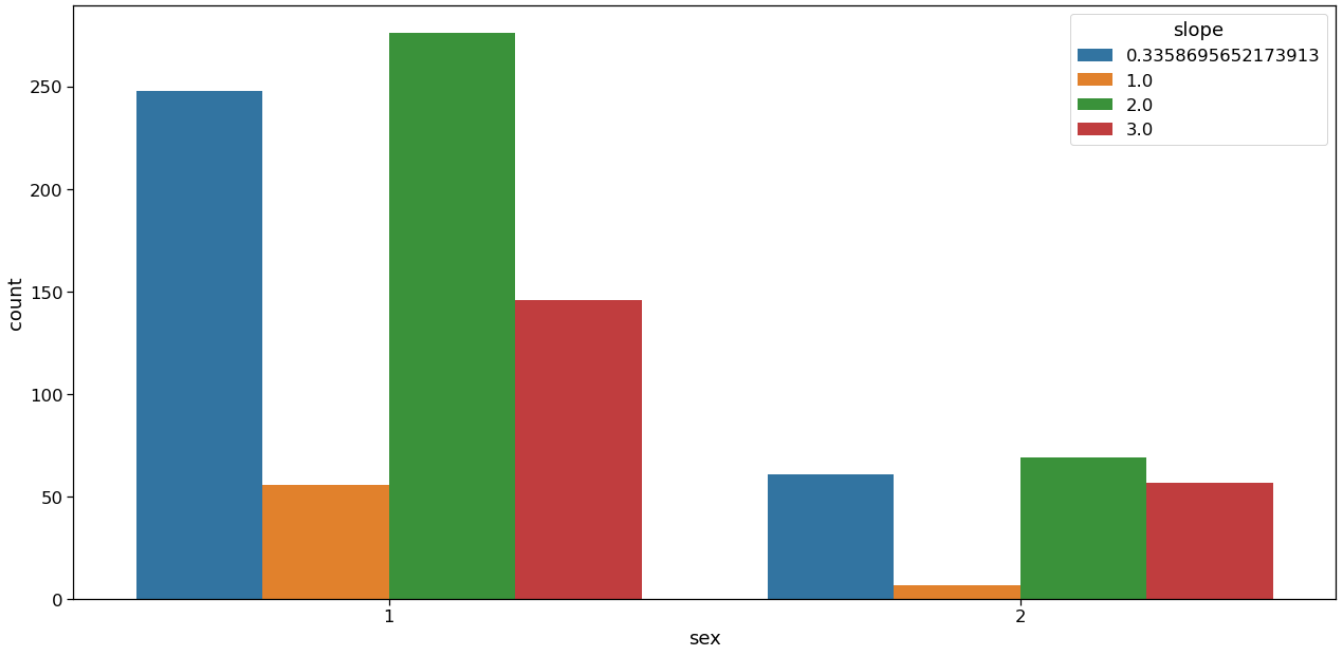
In [73]:
```
pyplot.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(df['sex'])
pyplot.tight_layout()
```

```
C:\Users\moham\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



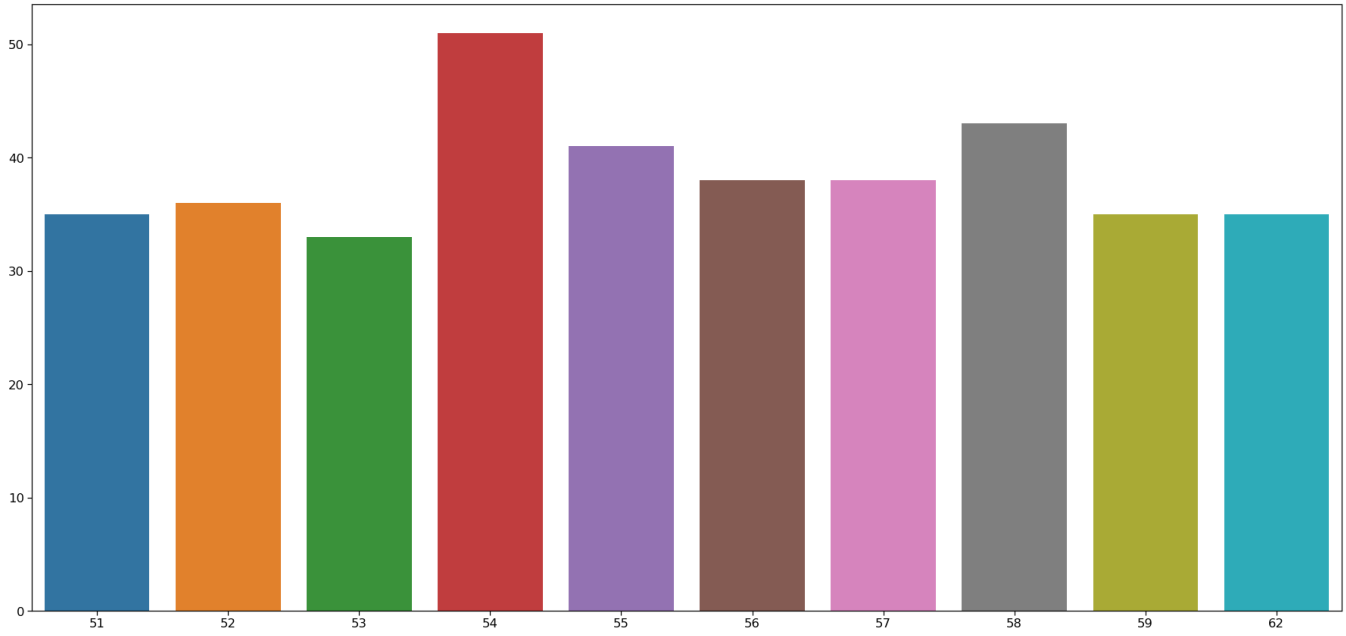
In [74]:
```
pyplot.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(df['sex'],hue=df["slope"])
pyplot.tight_layout()
```

```
C:\Users\moham\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



Age analysis

In [75]:
```
pyplot.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=df.age.value_counts()[:10].index,y=df.age.value_counts()[:10].values)
pyplot.tight_layout()
```



54 column has the highest ferquency

checking the range of age in the dataset

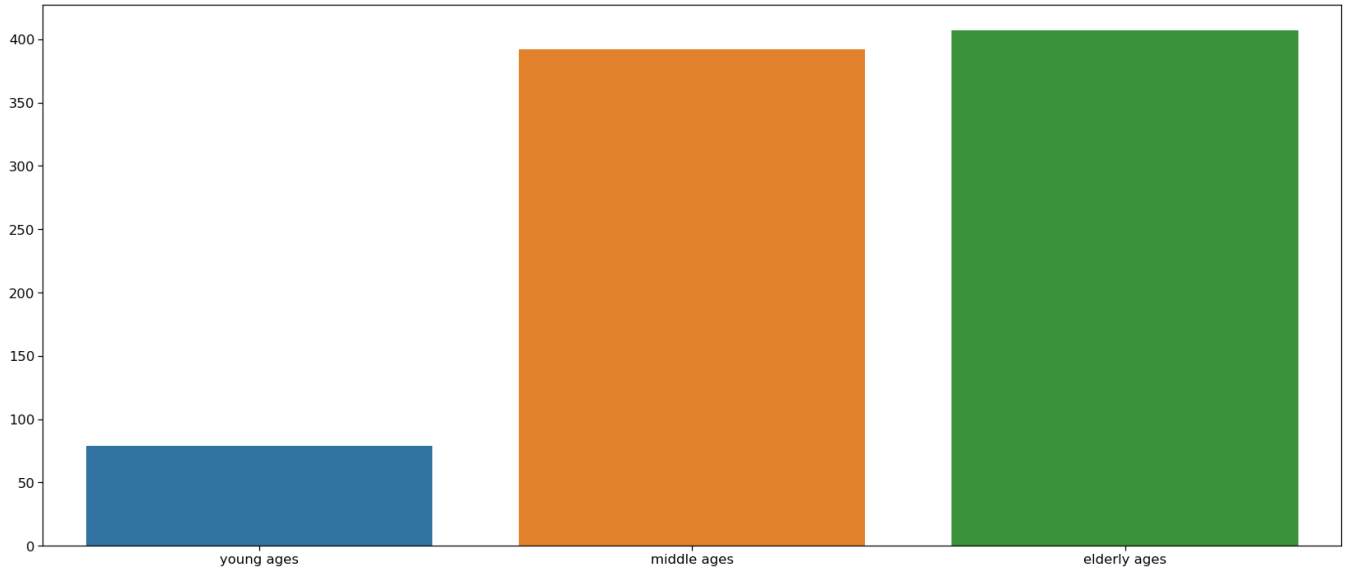
In [76]:
```
minAge=min(df.age)
maxAge=max(df.age)
meanAge=df.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
```

```
Min Age : 28
Max Age : 77
Mean Age : 53.51086956521739
```

Dividing the age feature into Young , Middle, Old

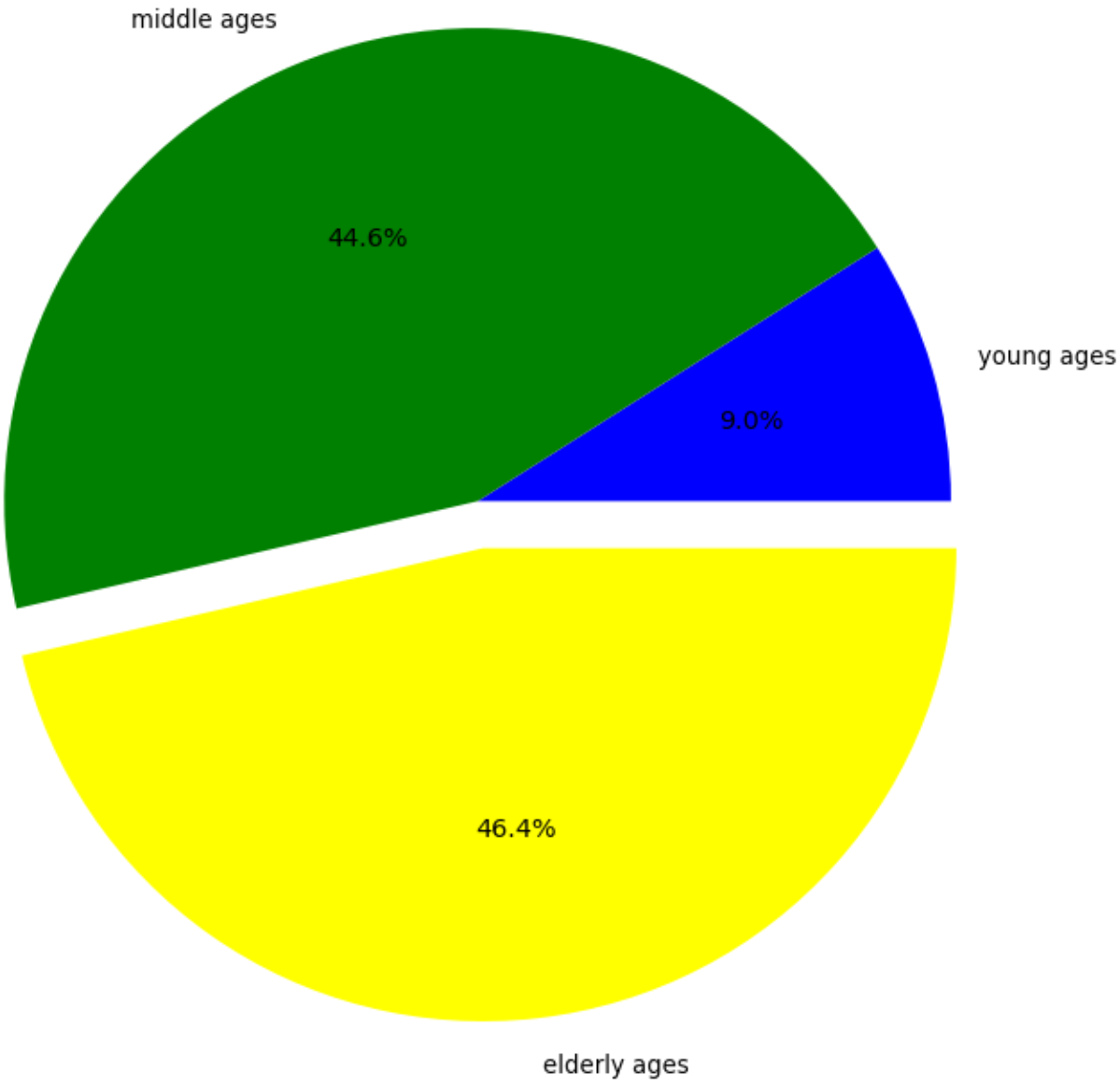
In [77]:
```
Young = df[(df.age>=29)&(df.age<40)]
Middle = df[(df.age>=40)&(df.age<55)]
Elder = df[(df.age>55)]

pyplot.figure(figsize=(23,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder
pyplot.tight_layout()
```



pie plot

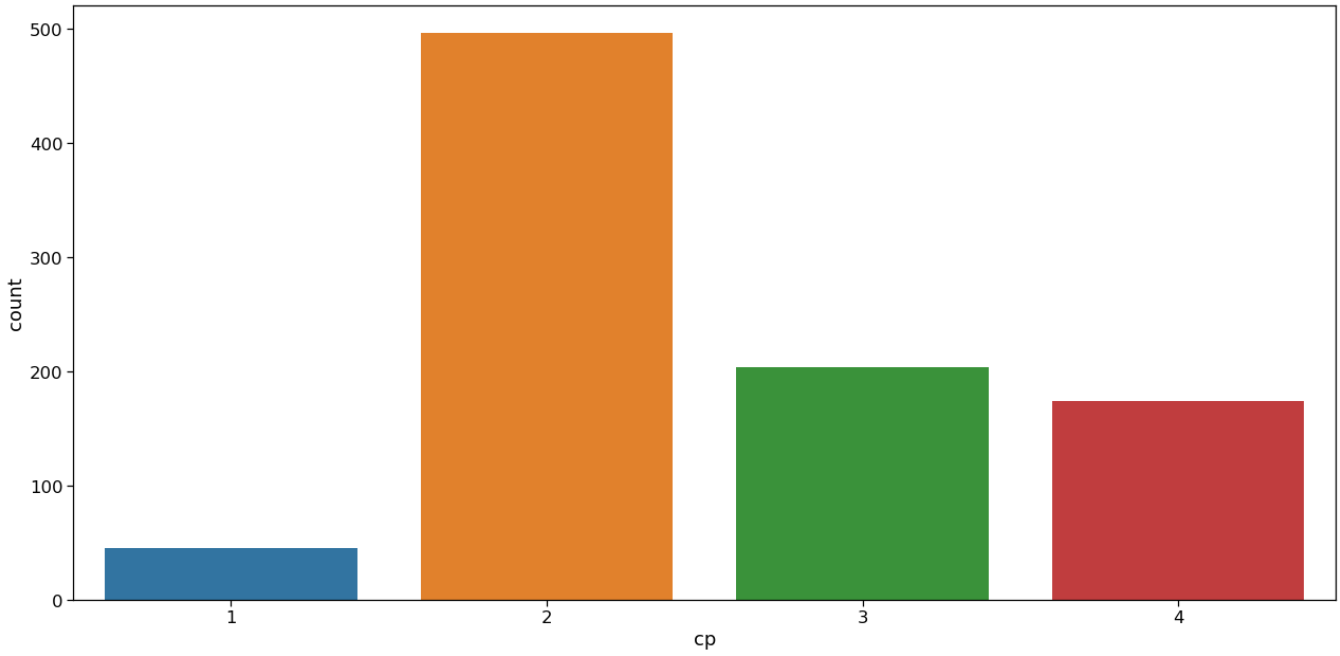
In [78]:
```
colors = ['blue','green','yellow']
explode = [0,0,0.1]
pyplot.figure(figsize=(10,10))
sns.set_context('notebook',font_scale = 1.2)
pyplot.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle ages','elderly ag
pyplot.tight_layout()
```



Chest Pain Type("cp") Analysis

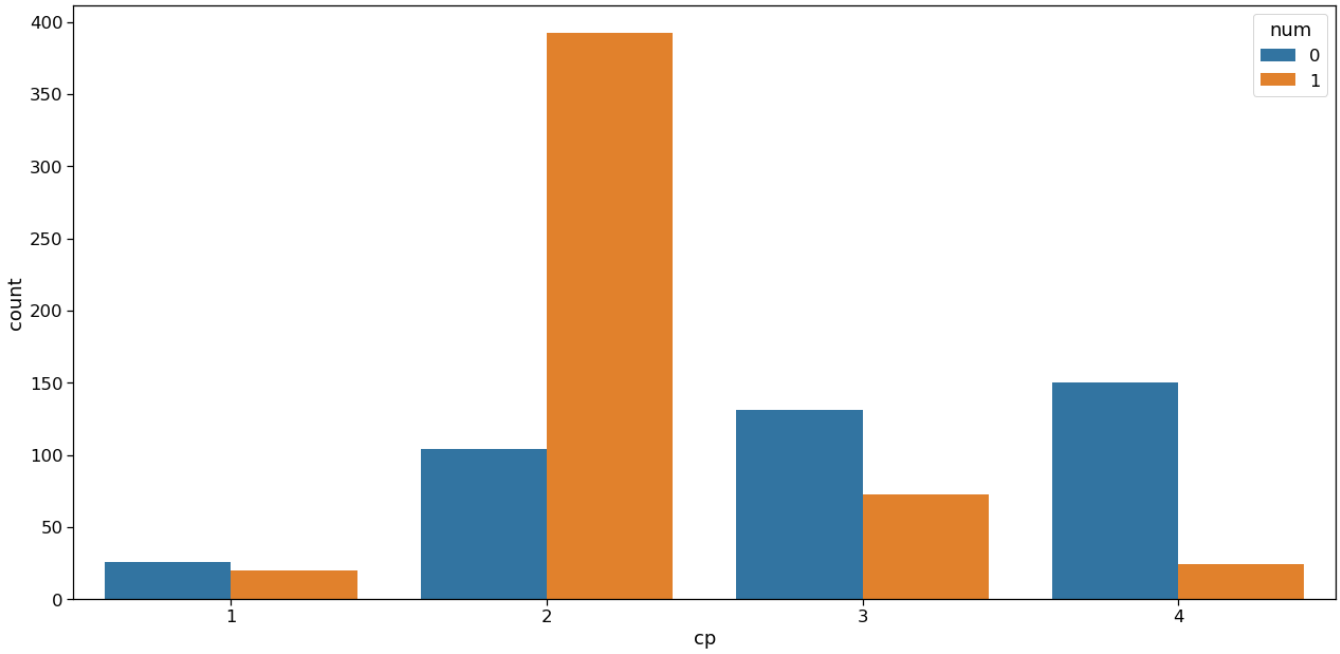
In [79]:
```
pyplot.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(df['cp'])
pyplot.tight_layout()
```

```
C:\Users\moham\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



In [80]:
```
pyplot.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(df['cp'],hue=df['num'])
pyplot.tight_layout()
```

```
C:\Users\moham\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Feature Engineering

In [81]:
```
categorical_val = []
continous_val = []
for column in df.columns:
    print("--------------------")
    print(f"{column} : {df[column].unique()}")
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

```
--------------------
id : [  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
 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504
 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522
 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540
 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558
 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576
 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594
 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612
 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630
 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648
 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666
 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684
 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702
 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720
 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738
 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756
 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774
 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792
 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810
 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828
 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846
 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864
 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882
 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900
 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918
 919 920]
--------------------
age : [63 67 37 41 56 62 57 53 44 52 48 54 49 64 58 60 50 66 43 40 69 59 42 55
 61 65 71 51 46 45 39 68 47 34 35 29 70 77 38 74 76 28 30 31 32 33 36 72
 73 75]
--------------------
sex : [1 2]
--------------------
dataset : [1 2 3 4]
--------------------
cp : [1 2 3 4]
--------------------
trestbps : [1.45000000e+02 1.60000000e+02 1.20000000e+02 1.30000000e+02
 1.40000000e+02 1.72000000e+02 1.50000000e+02 1.10000000e+02
 1.32000000e+02 1.17000000e+02 1.35000000e+02 1.12000000e+02
 1.05000000e+02 1.24000000e+02 1.25000000e+02 1.42000000e+02
 1.28000000e+02 1.70000000e+02 1.55000000e+02 1.04000000e+02
```

```
 1.80000000e+02 1.38000000e+02 1.08000000e+02 1.34000000e+02
 1.22000000e+02 1.15000000e+02 1.18000000e+02 1.00000000e+02
 2.00000000e+02 9.40000000e+01 1.65000000e+02 1.02000000e+02
 1.52000000e+02 1.01000000e+02 1.26000000e+02 1.74000000e+02
 1.48000000e+02 1.78000000e+02 1.58000000e+02 1.92000000e+02
 1.29000000e+02 1.44000000e+02 1.23000000e+02 1.36000000e+02
 1.46000000e+02 1.06000000e+02 1.56000000e+02 1.54000000e+02
 1.14000000e+02 1.64000000e+02 9.80000000e+01 1.90000000e+02
 6.41304348e-02 1.13000000e+02 9.20000000e+01 9.50000000e+01
 8.00000000e+01 1.85000000e+02 1.16000000e+02 0.00000000e+00
 9.60000000e+01 1.27000000e+02]
--------------------
chol : [2.33000000e+02 2.86000000e+02 2.29000000e+02 2.50000000e+02
 2.04000000e+02 2.36000000e+02 2.68000000e+02 3.54000000e+02
 2.54000000e+02 2.03000000e+02 1.92000000e+02 2.94000000e+02
 2.56000000e+02 2.63000000e+02 1.99000000e+02 1.68000000e+02
 2.39000000e+02 2.75000000e+02 2.66000000e+02 2.11000000e+02
 2.83000000e+02 2.84000000e+02 2.24000000e+02 2.06000000e+02
 2.19000000e+02 3.40000000e+02 2.26000000e+02 2.47000000e+02
 1.67000000e+02 2.30000000e+02 3.35000000e+02 2.34000000e+02
 1.77000000e+02 2.76000000e+02 3.53000000e+02 2.43000000e+02
 2.25000000e+02 3.02000000e+02 2.12000000e+02 3.30000000e+02
 1.75000000e+02 4.17000000e+02 1.97000000e+02 1.98000000e+02
 2.90000000e+02 2.53000000e+02 1.72000000e+02 2.73000000e+02
 2.13000000e+02 3.05000000e+02 2.16000000e+02 3.04000000e+02
 1.88000000e+02 2.82000000e+02 1.85000000e+02 2.32000000e+02
 3.26000000e+02 2.31000000e+02 2.69000000e+02 2.67000000e+02
 2.48000000e+02 3.60000000e+02 2.58000000e+02 3.08000000e+02
 2.45000000e+02 2.70000000e+02 2.08000000e+02 2.64000000e+02
 3.21000000e+02 2.74000000e+02 3.25000000e+02 2.35000000e+02
 2.57000000e+02 1.64000000e+02 1.41000000e+02 2.52000000e+02
 2.55000000e+02 2.01000000e+02 2.22000000e+02 2.60000000e+02
 1.82000000e+02 3.03000000e+02 2.65000000e+02 3.09000000e+02
 3.07000000e+02 2.49000000e+02 1.86000000e+02 3.41000000e+02
 1.83000000e+02 4.07000000e+02 2.17000000e+02 2.88000000e+02
 2.20000000e+02 2.09000000e+02 2.27000000e+02 2.61000000e+02
 1.74000000e+02 2.81000000e+02 2.21000000e+02 2.05000000e+02
 2.40000000e+02 2.89000000e+02 3.18000000e+02 2.98000000e+02
 5.64000000e+02 2.46000000e+02 3.22000000e+02 2.99000000e+02
 3.00000000e+02 2.93000000e+02 2.77000000e+02 2.14000000e+02
 2.07000000e+02 2.23000000e+02 1.60000000e+02 3.94000000e+02
 1.84000000e+02 3.15000000e+02 4.09000000e+02 2.44000000e+02
 1.95000000e+02 1.96000000e+02 1.26000000e+02 3.13000000e+02
 2.59000000e+02 2.00000000e+02 2.62000000e+02 2.15000000e+02
 2.28000000e+02 1.93000000e+02 2.71000000e+02 2.10000000e+02
 3.27000000e+02 1.49000000e+02 2.95000000e+02 3.06000000e+02
 1.78000000e+02 2.37000000e+02 2.18000000e+02 2.42000000e+02
 3.19000000e+02 1.66000000e+02 1.80000000e+02 3.11000000e+02
 2.78000000e+02 3.42000000e+02 1.69000000e+02 1.87000000e+02
 1.57000000e+02 1.76000000e+02 2.41000000e+02 1.31000000e+02
 1.32000000e+02 3.26086957e-02 1.61000000e+02 1.73000000e+02
 1.94000000e+02 2.97000000e+02 2.92000000e+02 3.39000000e+02
 1.47000000e+02 2.91000000e+02 3.58000000e+02 4.12000000e+02
 2.38000000e+02 1.63000000e+02 2.80000000e+02 2.02000000e+02
 3.28000000e+02 1.29000000e+02 1.90000000e+02 1.79000000e+02
 2.72000000e+02 1.00000000e+02 4.68000000e+02 3.20000000e+02
 3.12000000e+02 1.71000000e+02 3.65000000e+02 3.44000000e+02
 8.50000000e+01 3.47000000e+02 2.51000000e+02 2.87000000e+02
 1.56000000e+02 1.17000000e+02 4.66000000e+02 3.38000000e+02
 5.29000000e+02 3.92000000e+02 3.29000000e+02 3.55000000e+02
 6.03000000e+02 4.04000000e+02 5.18000000e+02 2.85000000e+02
 2.79000000e+02 3.88000000e+02 3.36000000e+02 4.91000000e+02
 3.31000000e+02 3.93000000e+02 0.00000000e+00 1.53000000e+02
 3.16000000e+02 4.58000000e+02 3.84000000e+02 3.49000000e+02
 1.42000000e+02 1.81000000e+02 3.10000000e+02 1.70000000e+02
 3.69000000e+02 1.65000000e+02 3.37000000e+02 3.33000000e+02
 1.39000000e+02 3.85000000e+02]
--------------------
fbs : [True False 0.09782608695652174]
```

```
--------------------
restecg : [1.00000000e+00 2.00000000e+00 3.00000000e+00 2.17391304e-03]
--------------------
thalch : [1.50000000e+02 1.08000000e+02 1.29000000e+02 1.87000000e+02
 1.72000000e+02 1.78000000e+02 1.60000000e+02 1.63000000e+02
 1.47000000e+02 1.55000000e+02 1.48000000e+02 1.53000000e+02
 1.42000000e+02 1.73000000e+02 1.62000000e+02 1.74000000e+02
 1.68000000e+02 1.39000000e+02 1.71000000e+02 1.44000000e+02
 1.32000000e+02 1.58000000e+02 1.14000000e+02 1.51000000e+02
 1.61000000e+02 1.79000000e+02 1.20000000e+02 1.12000000e+02
 1.37000000e+02 1.57000000e+02 1.69000000e+02 1.65000000e+02
 1.23000000e+02 1.28000000e+02 1.52000000e+02 1.40000000e+02
 1.88000000e+02 1.09000000e+02 1.25000000e+02 1.31000000e+02
 1.70000000e+02 1.13000000e+02 9.90000000e+01 1.77000000e+02
 1.41000000e+02 1.80000000e+02 1.11000000e+02 1.43000000e+02
 1.82000000e+02 1.56000000e+02 1.15000000e+02 1.49000000e+02
 1.45000000e+02 1.46000000e+02 1.75000000e+02 1.86000000e+02
 1.85000000e+02 1.59000000e+02 1.30000000e+02 1.90000000e+02
 1.36000000e+02 9.70000000e+01 1.27000000e+02 1.54000000e+02
 1.33000000e+02 1.26000000e+02 2.02000000e+02 1.03000000e+02
 1.66000000e+02 1.64000000e+02 1.84000000e+02 1.24000000e+02
 1.22000000e+02 9.60000000e+01 1.38000000e+02 8.80000000e+01
 1.05000000e+02 1.94000000e+02 1.95000000e+02 1.06000000e+02
 1.67000000e+02 9.50000000e+01 1.92000000e+02 1.17000000e+02
 1.21000000e+02 1.16000000e+02 7.10000000e+01 1.18000000e+02
 1.81000000e+02 1.34000000e+02 9.00000000e+01 9.80000000e+01
 1.76000000e+02 1.35000000e+02 1.10000000e+02 5.97826087e-02
 1.00000000e+02 8.70000000e+01 1.02000000e+02 9.20000000e+01
 9.10000000e+01 8.20000000e+01 1.19000000e+02 9.40000000e+01
 1.04000000e+02 6.00000000e+01 8.30000000e+01 6.30000000e+01
 7.00000000e+01 7.70000000e+01 7.20000000e+01 7.80000000e+01
 8.60000000e+01 9.30000000e+01 6.70000000e+01 8.40000000e+01
 8.00000000e+01 1.07000000e+02 6.90000000e+01 7.30000000e+01]
--------------------
exang : [False True 0.059782608695652176]
--------------------
oldpeak : [ 2.3        1.5        2.6        3.5        1.4        0.8
  3.6        0.6        3.1        0.4        1.3        0.
  0.5        1.6        1.         1.2        0.2        1.8
  3.2        2.4        2.         2.5        2.2        2.8
  3.         3.4        6.2        4.         5.6        2.9
  0.1        2.1        1.9        4.2        0.9        1.1
  3.8        0.7        0.3        4.4        5.         0.0673913
 -1.1       -1.5       -0.1       -2.6       -0.7       -2.
 -1.         1.7       -0.8       -0.5       -0.9        3.7      ]
--------------------
slope : [1.         2.         3.         0.33586957]
--------------------
ca : [0.         3.         2.         1.         0.66413043]
--------------------
thal : [1.         2.         3.         0.52826087]
--------------------
num : [0 1]
```

In [82]:

```
categorical_val.remove('num')
dfs = pd.get_dummies(df, columns = categorical_val)
dfs.head(6)
```

Out[82]:

	id	age	trestbps	chol	thalch	oldpeak	num	sex_1	sex_2	dataset_1	...	slope_3.0	ca_0.0	ca_0.664130
0	1	63	145.0	233.0	150.0	2.3	0	1	0	1	...	0	1	
1	2	67	160.0	286.0	108.0	1.5	1	1	0	1	...	0	0	
2	3	67	120.0	229.0	129.0	2.6	1	1	0	1	...	0	0	
3	4	37	130.0	250.0	187.0	3.5	0	1	0	1	...	0	1	
4	5	41	130.0	204.0	172.0	1.4	0	0	1	1	...	1	1	
5	6	56	120.0	236.0	178.0	0.8	0	1	0	1	...	1	1	

6 rows × 40 columns

Standard Scalar

In [84]:
```
sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak']
dfs[col_to_scale] = sc.fit_transform(dfs[col_to_scale])
dfs.head(6)
```

Out[84]:

	id	age	trestbps	chol	thalch	oldpeak	num	sex_1	sex_2	dataset_1	...	slope_3.0	ca_0.0
0	1	1.007386	0.573001	0.352519	0.502335	1.375912	0	1	0	1	...	0	1
1	2	1.432034	0.975820	0.815417	-0.518197	0.630106	1	1	0	1	...	0	0
2	3	1.432034	-0.098363	0.317583	-0.007931	1.655590	1	1	0	1	...	0	0
3	4	-1.752828	0.170183	0.500996	1.401375	2.494621	0	1	0	1	...	0	1
4	5	-1.328180	0.170183	0.099235	1.036900	0.536881	0	0	1	1	...	1	1
5	6	0.264251	-0.098363	0.378721	1.182690	-0.022474	0	1	0	1	...	1	1

6 rows × 40 columns

Splitting the Dataset

In [85]:
```
X = dfs.drop('num', axis=1)
y = dfs.num

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

KNN

In [86]:
```
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train,y_train)
y_pred1 = knn.predict(X_test)
print(accuracy_score(y_test,y_pred1))
```

```
0.7898550724637681
```

In [ ]: