

# MANOJ KUMAR - 2048015

## P12,P13: Hiarchical Clustering & DBSCAN

Compare and contrast the working of Hiarchical Clustering & DBSCAN on the dataset relevant to your domain.

### Importing basic libraries

```
In [38]: import pandas as pd
import numpy as np
import time
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
from dataprep.eda import create_report
```

### Reading the dataset

```
In [39]: ckd_df = pd.read_csv('kidney_disease.csv')

#Check the shape
print(ckd_df.shape)

(400, 26)
```

```
In [40]: #check the columns
ckd_df.columns
```

```
Out[40]: Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
               'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
               'appet', 'pe', 'ane', 'classification'],
              dtype='object')
```

### Rename the columns to have meaningful names

```
In [41]: col_dict={"bp": "blood_pressure",
                  "sg": "specific_gravity",
                  "al": "albumin",
                  "su": "sugar",
                  "rbc": "red_blood_cells",
                  "pc": "pus_cell",
                  "pcc": "pus_cell_clumps",
                  "ba": "bacteria",
                  "bgr": "blood_glucose_random",
                  "bu": "blood_urea",
                  "sc": "serum_creatinine",
                  "sod": "sodium",
                  "pot": "potassium",
                  "hemo": "hemoglobin",
                  "pcv": "packed_cell_volume",
                  "wc": "white_blood_cell_count",
                  "rc": "red_blood_cell_count",
                  "htn": "hypertension",
                  "dm": "diabetes_mellitus",
                  "cad": "coronary_artery_disease",
                  "appet": "appetite",
                  "pe": "pedal_edema",
                  "ane": "anemia"}

ckd_df.rename(columns=col_dict, inplace=True)

#Check the column names again
ckd_df.columns
```

```
Out[41]: Index(['id', 'age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
               'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
               'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium',
               'potassium', 'hemoglobin', 'packed_cell_volume',
               'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
               'diabetes_mellitus', 'coronary_artery_disease', 'appetite',
               'pedal_edema', 'anemia', 'classification'],
              dtype='object')
```

### Observing the data

```
In [42]: ckd_df.head(11).T
```

Out[42]:

	0	1	2	3	4	5	6	7	8	9	10
id	0	1	2	3	4	5	6	7	8	9	10
age	48	7	62	48	51	60	68	24	52	53	50
blood_pressure	80	50	80	70	80	90	70	NaN	100	90	60
specific_gravity	1.02	1.02	1.01	1.005	1.01	1.015	1.01	1.015	1.015	1.02	1.01
albumin	1	4	2	4	2	3	0	2	3	2	2
sugar	0	0	3	0	0	0	0	4	0	0	4
red_blood_cells	NaN	NaN	normal	normal	normal	NaN	NaN	normal	normal	abnormal	NaN
pus_cell	normal	normal	normal	abnormal	normal	NaN	normal	abnormal	abnormal	abnormal	abnormal
pus_cell_clumps	notpresent	notpresent	notpresent	present	notpresent	notpresent	notpresent	notpresent	present	present	present
bacteria	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent
blood_glucose_random	121	NaN	423	117	106	74	100	410	138	70	490
blood_urea	36	18	53	56	26	25	54	31	60	107	55
serum_creatinine	1.2	0.8	1.8	3.8	1.4	1.1	24	1.1	1.9	7.2	4
sodium	NaN	NaN	NaN	111	NaN	142	104	NaN	NaN	114	NaN
potassium	NaN	NaN	NaN	2.5	NaN	3.2	4	NaN	NaN	3.7	NaN
hemoglobin	15.4	11.3	9.6	11.2	11.6	12.2	12.4	12.4	10.8	9.5	9.4
packed_cell_volume	44	38	31	32	35	39	36	44	33	29	28
white_blood_cell_count	7800	6000	7500	6700	7300	7800	NaN	6900	9600	12100	NaN
red_blood_cell_count	5.2	NaN	NaN	3.9	4.6	4.4	NaN	5	4.0	3.7	NaN
hypertension	yes	no	no	yes	no	yes	no	no	yes	yes	yes
diabetes_mellitus	yes	no	yes	no	no	yes	no	yes	yes	yes	yes
coronary_artery_disease	no	no	no	no	no	no	no	no	no	no	no
appetite	good	good	poor	poor	good	good	good	good	good	poor	good
pedal_edema	no	no	no	yes	no	yes	no	yes	no	no	no
anemia	no	no	yes	yes	no	no	no	no	yes	yes	yes
classification	ckd	ckd	ckd	ckd	ckd	ckd	ckd	ckd	ckd	ckd	ckd

Data DeepDive

```
In [43]: for i in ckd_df.drop("id",axis=1).columns:
          print('unique values in "{}":\n'.format(i),ckd_df[i].unique())
```

```
unique values in "age":
[48.  7. 62. 51. 60. 68. 24. 52. 53. 50. 63. 40. 47. 61. 21. 42. 75. 69.
 nan 73. 70. 65. 76. 72. 82. 46. 45. 35. 54. 11. 59. 67. 15. 55. 44. 26.
 64. 56.  5. 74. 38. 58. 71. 34. 17. 12. 43. 41. 57.  8. 39. 66. 81. 14.
 27. 83. 30.  4.  3.  6. 32. 80. 49. 90. 78. 19.  2. 33. 36. 37. 23. 25.
 20. 29. 28. 22. 79.]
unique values in "blood_pressure":
[ 80.  50.  70.  90.  nan 100.  60. 110. 140. 180. 120.]
unique values in "specific_gravity":
[1.02  1.01  1.005 1.015  nan 1.025]
unique values in "albumin":
[ 1.  4.  2.  3.  0. nan  5.]
unique values in "sugar":
[ 0.  3.  4.  1. nan  2.  5.]
unique values in "red_blood_cells":
[nan 'normal' 'abnormal']
unique values in "pus_cell":
['normal' 'abnormal' nan]
unique values in "pus_cell_clumps":
[notpresent' 'present' nan]
```

```

In [44]: #Replace incorrect values
ckd_df['diabetes_mellitus'] =ckd_df['diabetes_mellitus'].replace(to_replace={'\tno':'no','\tyes':'yes',' yes':''})
ckd_df['coronary_artery_disease'] = ckd_df['coronary_artery_disease'].replace(to_replace='\tno',value='no')
ckd_df['white_blood_cell_count'] = ckd_df['white_blood_cell_count'].replace(to_replace='\t8400',value='8400')
ckd_df["classification"]=ckd_df["classification"].replace("ckd\t", "ckd")

for i in range(ckd_df.shape[0]):
    if ckd_df.iloc[i,16]=='\t?':
        ckd_df.iloc[i,16]=np.nan
    if ckd_df.iloc[i,16]=='\t43':
        ckd_df.iloc[i,16]='43'
    if ckd_df.iloc[i,17]=='\t?':
        ckd_df.iloc[i,17]=np.nan
    if ckd_df.iloc[i,17]=='\t6200':
        ckd_df.iloc[i,17]= '6200'
    if ckd_df.iloc[i,18]=='\t?':
        ckd_df.iloc[i,18]=np.nan
    if ckd_df.iloc[i,25]=='ckd':
        ckd_df.iloc[i,25]='1'
    if ckd_df.iloc[i,25]=='notckd':
        ckd_df.iloc[i,25]='0'

for i in ckd_df.drop("id",axis=1).columns:
    print('unique values in "{}":\n'.format(i),ckd_df[i].unique())

unique values in "age":
[48.  7. 62. 51. 60. 68. 24. 52. 53. 50. 63. 40. 47. 61. 21. 42. 75. 69.
 nan 73. 70. 65. 76. 72. 82. 46. 45. 35. 54. 11. 59. 67. 15. 55. 44. 26.
 64. 56.  5. 74. 38. 58. 71. 34. 17. 12. 43. 41. 57.  8. 39. 66. 81. 14.
 27. 83. 30.  4.  3.  6. 32. 80. 49. 90. 78. 19.  2. 33. 36. 37. 23. 25.
 20. 29. 28. 22. 79.]
unique values in "blood_pressure":
[ 80.  50.  70.  90.  nan 100.  60. 110. 140. 180. 120.]
unique values in "specific_gravity":
[1.02  1.01  1.005 1.015  nan 1.025]
unique values in "albumin":
[ 1.  4.  2.  3.  0. nan  5.]
unique values in "sugar":
[ 0.  3.  4.  1. nan  2.  5.]
unique values in "red_blood_cells":
[nan 'normal' 'abnormal']
unique values in "pus_cell":
['normal' 'abnormal' nan]
unique values in "pus_cell_clumps":
['notpresent' 'present' nan]
unique values in "bacteria":
['notpresent' 'present' nan]
unique values in "blood_glucose_random":
[121.  nan 423. 117. 106.  74. 100. 410. 138.  70. 490. 380. 208.  98.
 157.  76.  99. 114. 263. 173.  95. 108. 156. 264. 123.  93. 107. 159.
 140. 171. 270.  92. 137. 204.  79. 207. 124. 144.  91. 162. 246. 253.
 141. 182.  86. 150. 146. 425. 112. 250. 360. 163. 129. 133. 102. 158.
 165. 132. 104. 127. 415. 169. 251. 109. 280. 210. 219. 295.  94. 172.
 101. 298. 153.  88. 226. 143. 115.  89. 297. 233. 294. 323. 125.  90.
 308. 118. 224. 128. 122. 214. 213. 268. 256.  84. 105. 288. 139.  78.
 273. 242. 424. 303. 148. 160. 192. 307. 220. 447. 309.  22. 111. 261.
 215. 234. 131. 352.  80. 239. 110. 130. 184. 252. 113. 230. 341. 255.
 103. 238. 248. 120. 241. 269. 201. 203. 463. 176.  82. 119.  97.  96.
  81. 116. 134.  85.  83.  87.  75.]
unique values in "blood_urea":
[ 36.  18.  53.  56.  26.  25.  54.  31.  60. 107.  55.  72.
  86.  90. 162.  46.  87.  27. 148. 180. 163.  nan  50.  75.
  45.  28. 155.  33.  39. 153.  29.  65. 103.  70.  80.  20.
 202.  77.  89.  24.  17.  32. 114.  66.  38. 164. 142.  96.
 391.  15. 111.  73.  19.  92.  35.  16. 139.  48.  85.  98.
 186.  37.  47.  52.  82.  51. 106.  22. 217.  88. 118.  50.1
  71.  34.  40.  21. 219.  30. 125. 166.  49. 208. 176.  68.
 145. 165. 322.  23. 235. 132.  76.  42.  44.  41. 113.  1.5
 146.  58. 133. 137.  67. 115. 223.  98.6 158.  94.  74. 150.
  61.  57.  95. 191.  93. 241.  64.  79. 215. 309.  10. ]
unique values in "serum_creatinine":
[ 1.2  0.8  1.8  3.8  1.4  1.1 24.  1.9  7.2  4.  2.7  2.1
 4.6  4.1  9.6  2.2  5.2  1.3  1.6  3.9 76.  7.7  nan  2.4
 7.3  1.5  2.5  2.  3.4  0.7  1. 10.8  6.3  5.9  0.9  3.
 3.25 9.7  6.4  3.2 32.  0.6  6.1  3.3  6.7  8.5  2.8 15.
 2.9  1.7  3.6  5.6  6.5  4.4 10.2 11.5  0.5 12.2  5.3  9.2
13.8 16.9  6.  7.1 18.  2.3 13.  48.1 14.2 16.4  2.6  7.5
 4.3 18.1 11.8  9.3  6.8 13.5 12.8 11.9 12. 13.4 15.2 13.3
 0.4 ]
unique values in "sodium":
[ nan 111. 142. 104. 114. 131. 138. 135. 130. 141. 139.  4.5
 136. 129. 140. 132. 133. 134. 125. 163. 137. 128. 143. 127.
 146. 126. 122. 147. 124. 115. 145. 113. 120. 150. 144. ]
unique values in "potassium":
[ nan 2.5 3.2 4.  3.7 4.2 5.8 3.4 6.4 4.9 4.1 4.3 5.2 3.8
 4.6 3.9 4.7 5.9 4.8 4.4 6.6 39.  5.5 5.  3.5 3.6 7.6 2.9
 4.5 5.7 5.4 5.3 47.  6.3 5.1 5.6 3.  2.8 2.7 6.5 3.3]

```

```

unique values in "hemoglobin":
[15.4 11.3  9.6 11.2 11.6 12.2 12.4 10.8  9.5  9.4  9.7  9.8  5.6  7.6
 12.6 12.1 12.7 10.3  7.7 10.9  nan 11.1  9.9 12.5 12.9 10.1 12.  13.
  7.9  9.3 15.  10.   8.6 13.6 10.2 10.5  6.6 11.   7.5 15.6 15.2  4.8
  9.1  8.1 11.9 13.5  8.3  7.1 16.1 10.4  9.2  6.2 13.9 14.1  6.  11.8
 11.7 11.4 14.   8.2 13.2  6.1  8.  12.3  8.4 14.3  9.   8.7 10.6 13.1
 10.7  5.5  5.8  6.8  8.8  8.5 13.8 11.5  7.3 13.7 12.8 13.4  6.3  3.1
 17.  15.9 14.5 15.5 16.2 14.4 14.2 16.3 14.8 16.5 15.7 13.3 14.6 16.4
 16.9 16.  14.7 16.6 14.9 16.7 16.8 15.8 15.1 17.1 17.2 15.3 17.3 17.4
 17.7 17.8 17.5 17.6]
unique values in "packed_cell_volume":
['44' '38' '31' '32' '35' '39' '36' '33' '29' '28' nan '16' '24' '37' '30'
 '34' '40' '45' '27' '48' '52' '14' '22' '18' '42' '17' '46' '23' '19'
 '25' '41' '26' '15' '21' '43' '20' '47' '9' '49' '50' '53' '51' '54']
unique values in "white_blood_cell_count":
['7800' '6000' '7500' '6700' '7300' nan '6900' '9600' '12100' '4500'
 '12200' '11000' '3800' '11400' '5300' '9200' '6200' '8300' '8400' '10300'
 '9800' '9100' '7900' '6400' '8600' '18900' '21600' '4300' '8500' '11300'
 '7200' '7700' '14600' '6300' '7100' '11800' '9400' '5500' '5800' '13200'
 '12500' '5600' '7000' '11900' '10400' '10700' '12700' '6800' '6500'
 '13600' '10200' '9000' '14900' '8200' '15200' '5000' '16300' '12400'
 '10500' '4200' '4700' '10900' '8100' '9500' '2200' '12800' '11200'
 '19100' '12300' '16700' '2600' '26400' '8800' '7400' '4900' '8000'
 '12000' '15700' '4100' '5700' '11500' '5400' '10800' '9900' '5200' '5900'
 '9300' '9700' '5100' '6600']
unique values in "red_blood_cell_count":
['5.2' nan '3.9' '4.6' '4.4' '5' '4.0' '3.7' '3.8' '3.4' '2.6' '2.8' '4.3'
 '3.2' '3.6' '4' '4.1' '4.9' '2.5' '4.2' '4.5' '3.1' '4.7' '3.5' '6.0'
 '5.0' '2.1' '5.6' '2.3' '2.9' '2.7' '8.0' '3.3' '3.0' '3' '2.4' '4.8'
 '5.4' '6.1' '6.2' '6.3' '5.1' '5.8' '5.5' '5.3' '6.4' '5.7' '5.9' '6.5']
unique values in "hypertension":
['yes' 'no' nan]
unique values in "diabetes_mellitus":
['yes' 'no' nan]
unique values in "coronary_artery_disease":
['no' 'yes' nan]
unique values in "appetite":
['good' 'poor' nan]
unique values in "pedal_edema":
['no' 'yes' nan]
unique values in "anemia":
['no' 'yes' nan]
unique values in "classification":
['1' '0']

```

```

In [45]: # Observing the summarized information of data
ckd_df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
#   Column                               Non-Null Count  Dtype
---  -
0   id                                   400 non-null    int64
1   age                                  391 non-null    float64
2   blood_pressure                       388 non-null    float64
3   specific_gravity                     353 non-null    float64
4   albumin                              354 non-null    float64
5   sugar                                351 non-null    float64
6   red_blood_cells                      248 non-null    object
7   pus_cell                             335 non-null    object
8   pus_cell_clumps                      396 non-null    object
9   bacteria                             396 non-null    object
10  blood_glucose_random                 356 non-null    float64
11  blood_urea                           381 non-null    float64
12  serum_creatinine                     383 non-null    float64
13  sodium                               313 non-null    float64
14  potassium                             312 non-null    float64
15  hemoglobin                           348 non-null    float64
16  packed_cell_volume                   329 non-null    object
17  white_blood_cell_count               294 non-null    object
18  red_blood_cell_count                 269 non-null    object
19  hypertension                         398 non-null    object
20  diabetes_mellitus                    398 non-null    object
21  coronary_artery_disease              398 non-null    object
22  appetite                             399 non-null    object
23  pedal_edema                          399 non-null    object
24  anemia                               399 non-null    object
25  classification                       400 non-null    object
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB

```

```
In [46]: ckd_df.iloc[:, -1]=ckd_df.iloc[:, -1].astype('int64')
ckd_df.head(11).T
```

Out[46]:

	0	1	2	3	4	5	6	7	8	9	10
id	0	1	2	3	4	5	6	7	8	9	10
age	48	7	62	48	51	60	68	24	52	53	50
blood_pressure	80	50	80	70	80	90	70	NaN	100	90	60
specific_gravity	1.02	1.02	1.01	1.005	1.01	1.015	1.01	1.015	1.015	1.02	1.01
albumin	1	4	2	4	2	3	0	2	3	2	2
sugar	0	0	3	0	0	0	0	4	0	0	4
red_blood_cells	NaN	NaN	normal	normal	normal	NaN	NaN	normal	normal	abnormal	NaN
pus_cell	normal	normal	normal	abnormal	normal	NaN	normal	abnormal	abnormal	abnormal	abnormal
pus_cell_clumps	notpresent	notpresent	notpresent	present	notpresent	notpresent	notpresent	notpresent	present	present	present
bacteria	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent
blood_glucose_random	121	NaN	423	117	106	74	100	410	138	70	490
blood_urea	36	18	53	56	26	25	54	31	60	107	55
serum_creatinine	1.2	0.8	1.8	3.8	1.4	1.1	24	1.1	1.9	7.2	4
sodium	NaN	NaN	NaN	111	NaN	142	104	NaN	NaN	114	NaN
potassium	NaN	NaN	NaN	2.5	NaN	3.2	4	NaN	NaN	3.7	NaN
hemoglobin	15.4	11.3	9.6	11.2	11.6	12.2	12.4	12.4	10.8	9.5	9.4
packed_cell_volume	44	38	31	32	35	39	36	44	33	29	28
white_blood_cell_count	7800	6000	7500	6700	7300	7800	NaN	6900	9600	12100	NaN
red_blood_cell_count	5.2	NaN	NaN	3.9	4.6	4.4	NaN	5	4.0	3.7	NaN
hypertension	yes	no	no	yes	no	yes	no	no	yes	yes	yes
diabetes_mellitus	yes	no	yes	no	no	yes	no	yes	yes	yes	yes
coronary_artery_disease	no	no	no	no	no	no	no	no	no	no	no
appetite	good	good	poor	poor	good	good	good	good	good	poor	good
pedal_edema	no	no	no	yes	no	yes	no	yes	no	no	no
anemia	no	no	yes	yes	no	no	no	no	yes	yes	yes
classification	1	1	1	1	1	1	1	1	1	1	1

```
In [47]: print(ckd_df['packed_cell_volume'].unique())
print(ckd_df['white_blood_cell_count'].unique())
print(ckd_df['red_blood_cell_count'].unique())

['44' '38' '31' '32' '35' '39' '36' '33' '29' '28' nan '16' '24' '37' '30'
'34' '40' '45' '27' '48' '52' '14' '22' '18' '42' '17' '46' '23' '19'
'25' '41' '26' '15' '21' '43' '20' '47' '9' '49' '50' '53' '51' '54']
['7800' '6000' '7500' '6700' '7300' nan '6900' '9600' '12100' '4500'
'12200' '11000' '3800' '11400' '5300' '9200' '6200' '8300' '8400' '10300'
'9800' '9100' '7900' '6400' '8600' '18900' '21600' '4300' '8500' '11300'
'7200' '7700' '14600' '6300' '7100' '11800' '9400' '5500' '5800' '13200'
'12500' '5600' '7000' '11900' '10400' '10700' '12700' '6800' '6500'
'13600' '10200' '9000' '14900' '8200' '15200' '5000' '16300' '12400'
'10500' '4200' '4700' '10900' '8100' '9500' '2200' '12800' '11200'
'19100' '12300' '16700' '2600' '26400' '8800' '7400' '4900' '8000'
'12000' '15700' '4100' '5700' '11500' '5400' '10800' '9900' '5200' '5900'
'9300' '9700' '5100' '6600']
['5.2' nan '3.9' '4.6' '4.4' '5' '4.0' '3.7' '3.8' '3.4' '2.6' '2.8' '4.3'
'3.2' '3.6' '4' '4.1' '4.9' '2.5' '4.2' '4.5' '3.1' '4.7' '3.5' '6.0'
'5.0' '2.1' '5.6' '2.3' '2.9' '2.7' '8.0' '3.3' '3.0' '3' '2.4' '4.8'
'5.4' '6.1' '6.2' '6.3' '5.1' '5.8' '5.5' '5.3' '6.4' '5.7' '5.9' '6.5']
```

```
In [48]: mistyped=['packed_cell_volume','white_blood_cell_count','red_blood_cell_count']
for col in mistyped:
    ckd_df[col]=ckd_df[col].astype('float')

numeric=[]
for i in ckd_df.columns:
    if ckd_df[i].dtype=='float64':
        numeric.append(i)

numeric
```

```
Out[48]: ['age',
'blood_pressure',
'specific_gravity',
'albumin',
'sugar',
'blood_glucose_random',
'blood_urea',
'serum_creatinine',
'sodium',
'potassium',
'hemoglobin',
'packed_cell_volume',
'white_blood_cell_count',
'red_blood_cell_count']
```

```
In [49]: ckd_df.drop('id',axis=1,inplace=True)

categoricals=[]

for col in ckd_df.columns:
    if not col in numeric:
        categoricals.append(col)
categoricals.remove('classification')

categoricals
```

```
Out[49]: ['red_blood_cells',
'pus_cell',
'pus_cell_clumps',
'bacteria',
'hypertension',
'diabetes_mellitus',
'coronary_artery_disease',
'appetite',
'pedal_edema',
'anemia']
```

```
In [50]: import warnings
warnings.simplefilter('ignore')

import matplotlib.style as style
style.use('fivethirtyeight')
```

***Checking distribution of the numerical features***

```
In [51]: fig, axes = plt.subplots(nrows=7, ncols=2, figsize=(15,30))
fig.subplots_adjust(hspace=0.5)
fig.suptitle('Distributions of numerical Features')

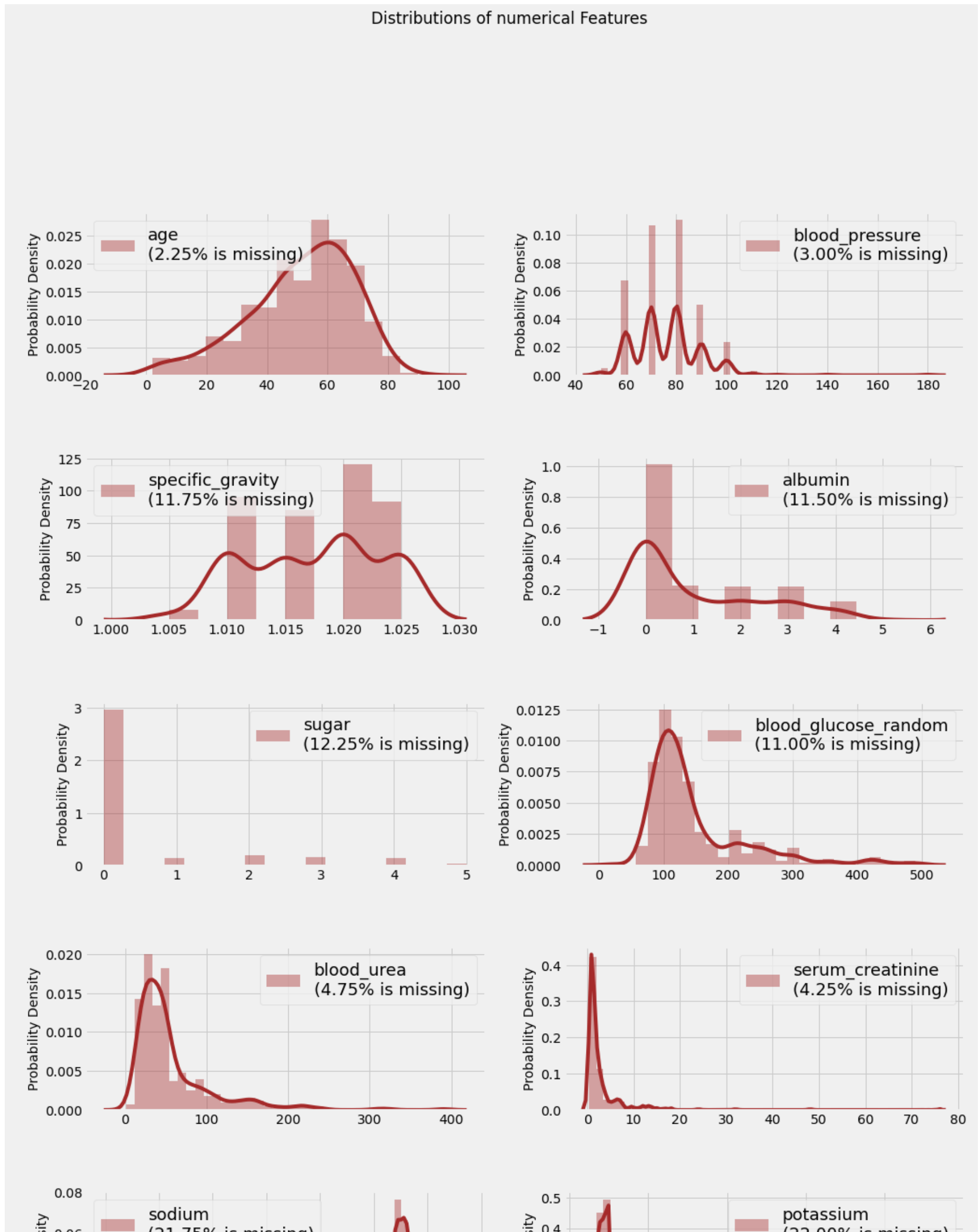
n_rows, n_cols = (7,2)

for index, column in enumerate(numeric):

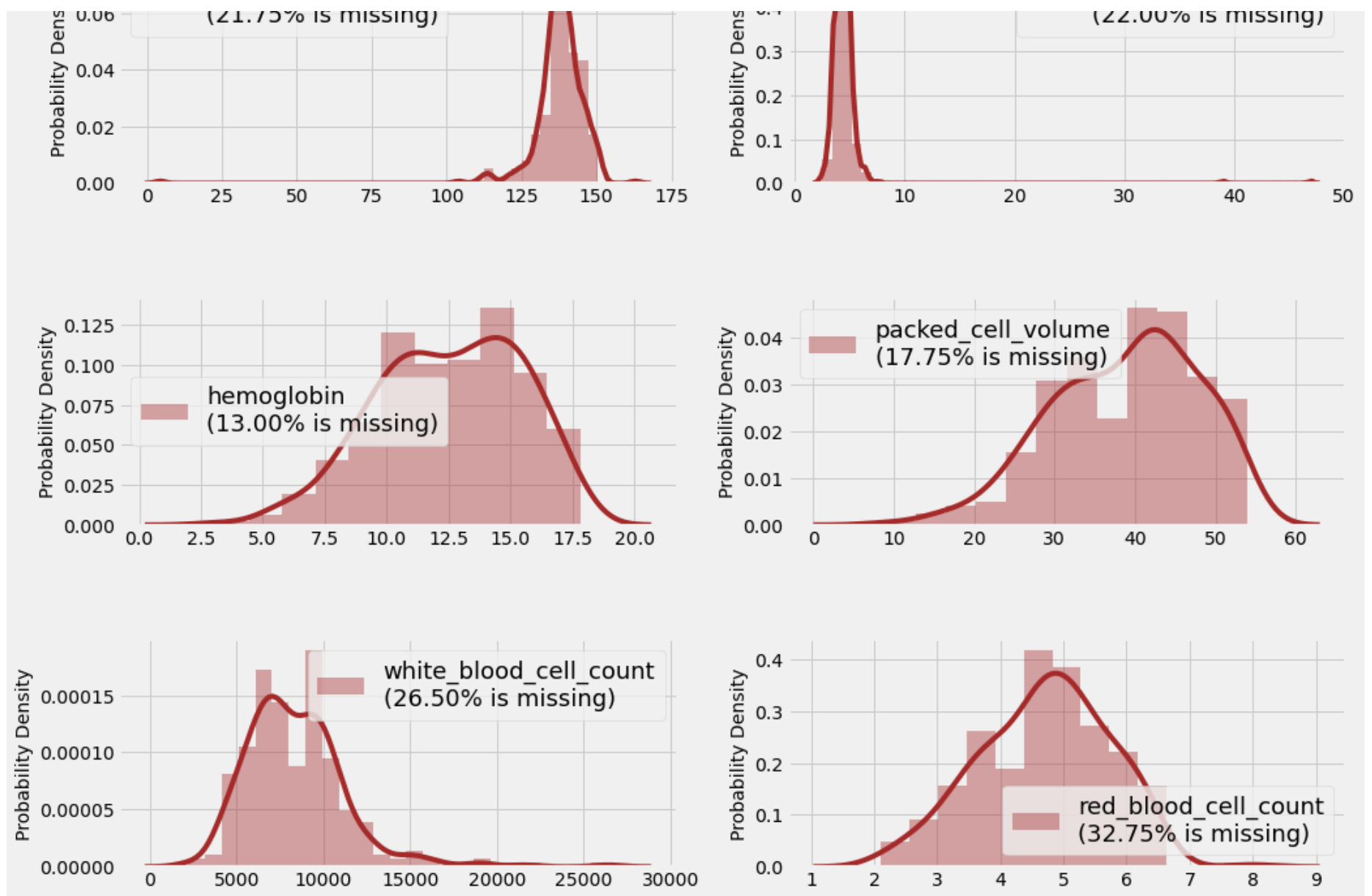
    i,j = (index // n_cols), (index % n_cols)
    miss_perc="%.2f"%(100*(1-(ckd_df[column].dropna().shape[0])/ckd_df.shape[0]))
    collabel=column+"\n({}% is missing)".format(miss_perc)
    fig=sns.distplot(ckd_df[column], color="brown", label=collabel,
                    norm_hist=True, ax=axes[i,j], kde_kws={"lw":4})
    fig=fig.legend(loc='best', fontsize=18)

    axes[i,j].set_ylabel("Probability Density",fontsize='medium')
    axes[i,j].set_xlabel(None)

plt.show()
```







**Checking distribution of the Categorical features**



```

In [52]: style.use('fivethirtyeight')

fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(15,30))
fig.subplots_adjust(hspace=0.5)
fig.suptitle('Distributions of Categorical Features')

n_rows, n_cols = (5,2)

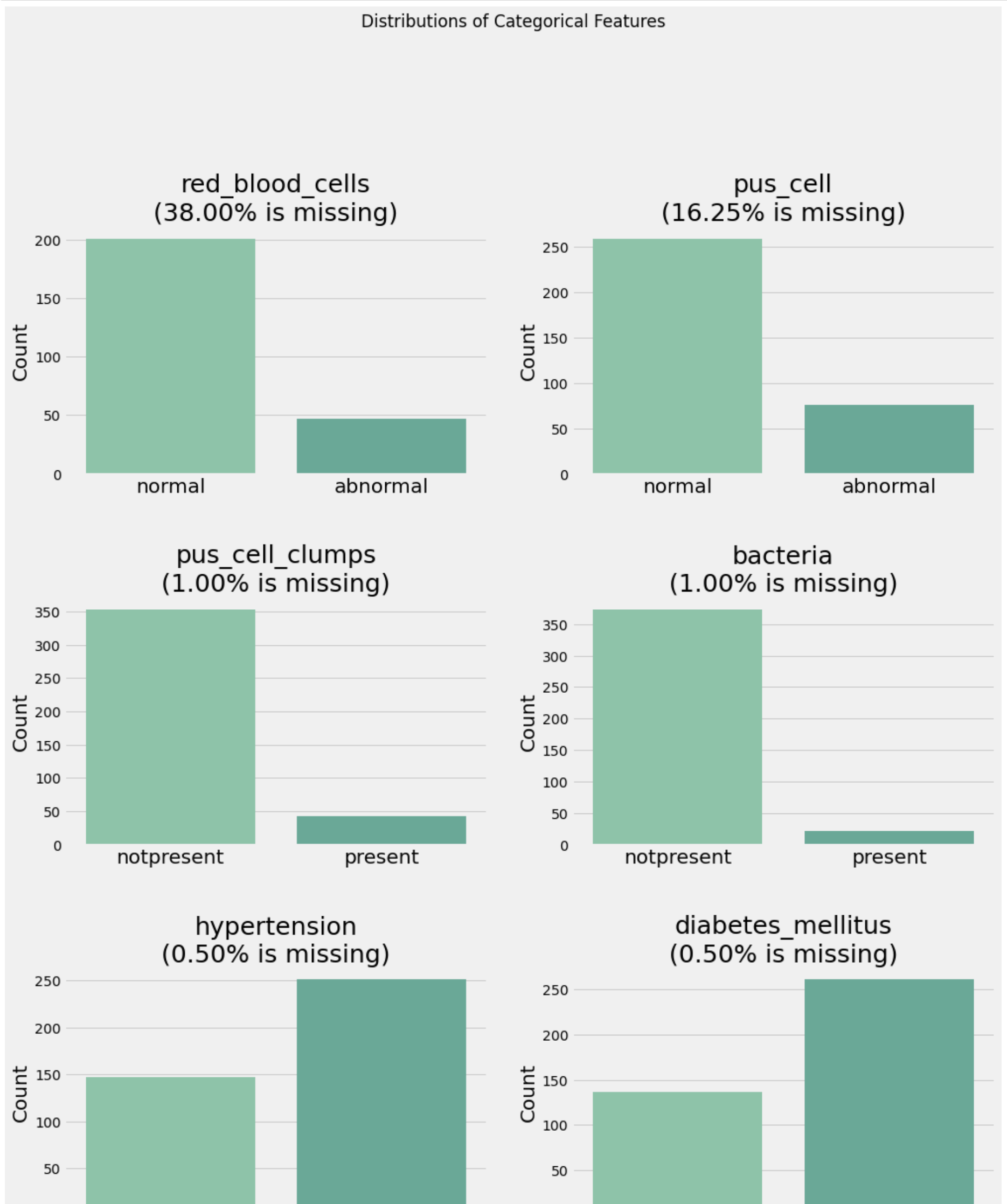
for index, column in enumerate(categoricals):

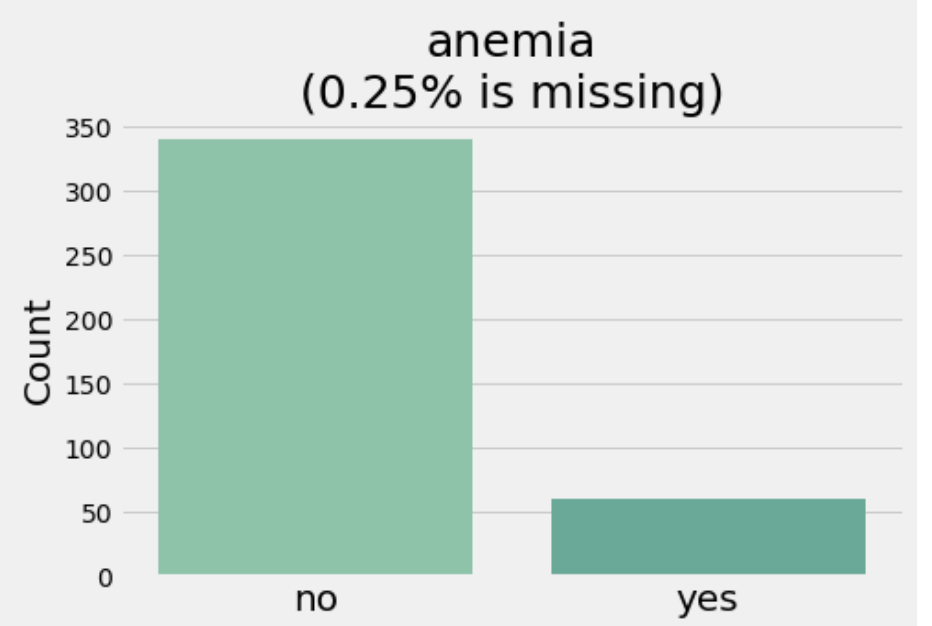
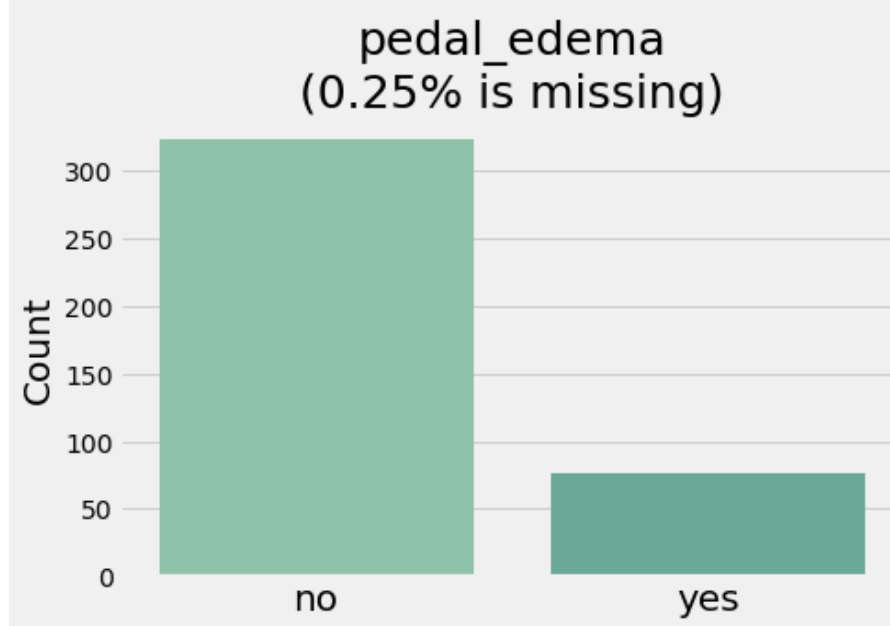
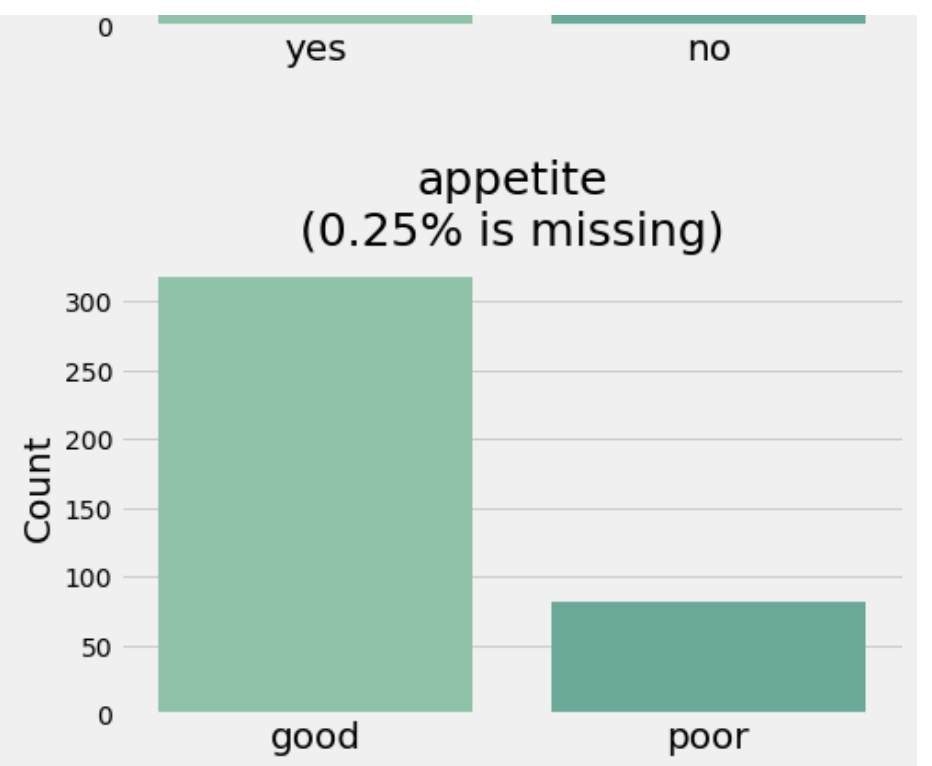
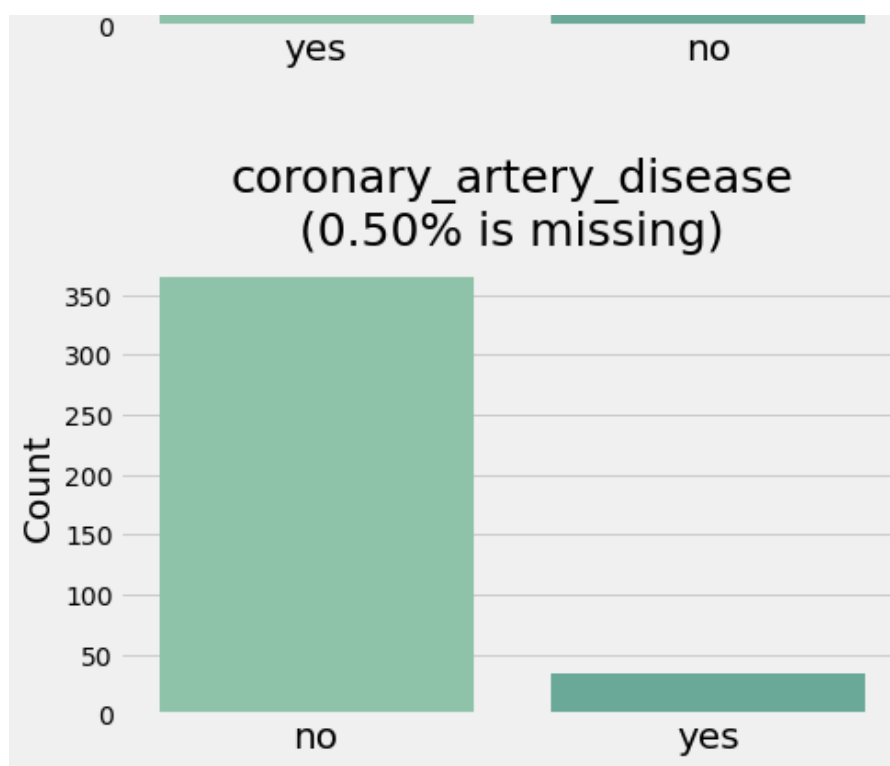
    i,j = index // n_cols, index % n_cols
    miss_perc="%.2f"%(100*(1-(ckd_df[column].dropna().shape[0])/ckd_df.shape[0]))
    collabel=column+"\n({}% is missing)".format(miss_perc)
    fig = sns.countplot(x=column, data=ckd_df,label=collabel,
                        palette=sns.cubehelix_palette(rot=-.4,light=0.7,hue=1), ax=axes[i,j])

    axes[i,j].set_title(collabel,fontsize=25)
    axes[i,j].set_xlabel(None)
    axes[i,j].set_ylabel("Count",fontsize=20)
    axes[i,j].set_xticklabels(axes[i,j].get_xticklabels(), Fontsize=20)

plt.show()

```





```
In [53]: import matplotlib.ticker as ticker
style.use('fivethirtyeight')
# Some random data
ncount = 400

plt.figure(figsize=(15,8))
ax = sns.countplot(x="classification", data=ckd_df)
plt.title('Distribution of classification sata')
plt.xlabel('Type')

# Make twin axis
ax2=ax.twinx()

# Switch so count axis is on right, frequency on left
ax2.yaxis.tick_left()
ax.yaxis.tick_right()

# Also switch the labels over
ax.yaxis.set_label_position('right')
ax2.yaxis.set_label_position('left')

ax2.set_ylabel('Frequency [%]')

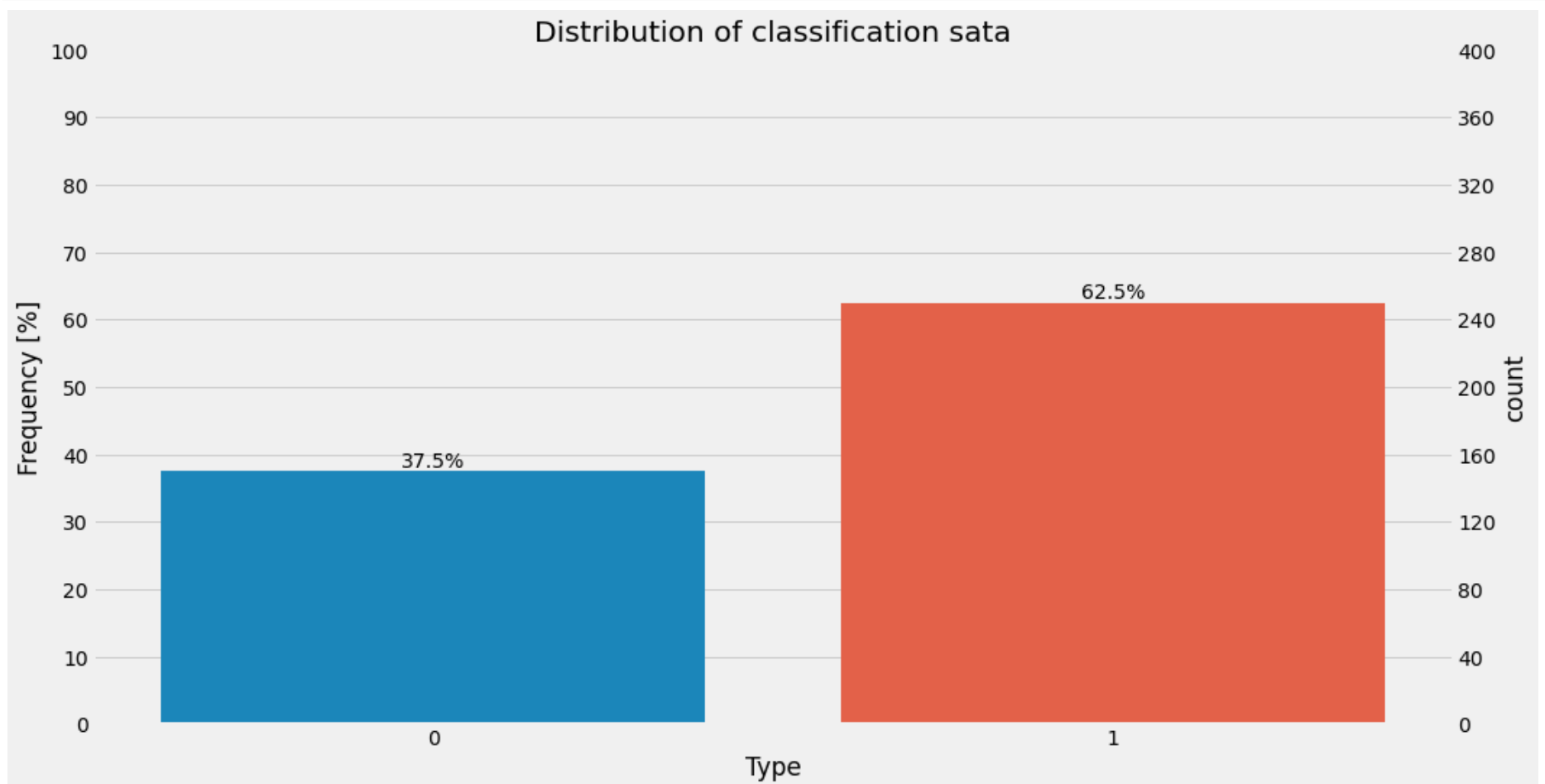
for p in ax.patches:
    x=p.get_bbox().get_points()[0,0]
    y=p.get_bbox().get_points()[1,1]
    ax.annotate('{:.1f}%'.format(100.*y/ncount), (x.mean(), y),
                ha='center', va='bottom') # set the alignment of the text

# Use a LinearLocator to ensure the correct number of ticks
ax.yaxis.set_major_locator(ticker.LinearLocator(11))

# Fix the frequency range to 0-100
ax2.set_ylim(0,100)
ax.set_ylim(0,ncount)

# And use a MultipleLocator to ensure a tick spacing of 10
ax2.yaxis.set_major_locator(ticker.MultipleLocator(10))

# Need to turn the grid on ax2 off, otherwise the gridlines end up on top of the bars
ax2.grid(None)
```



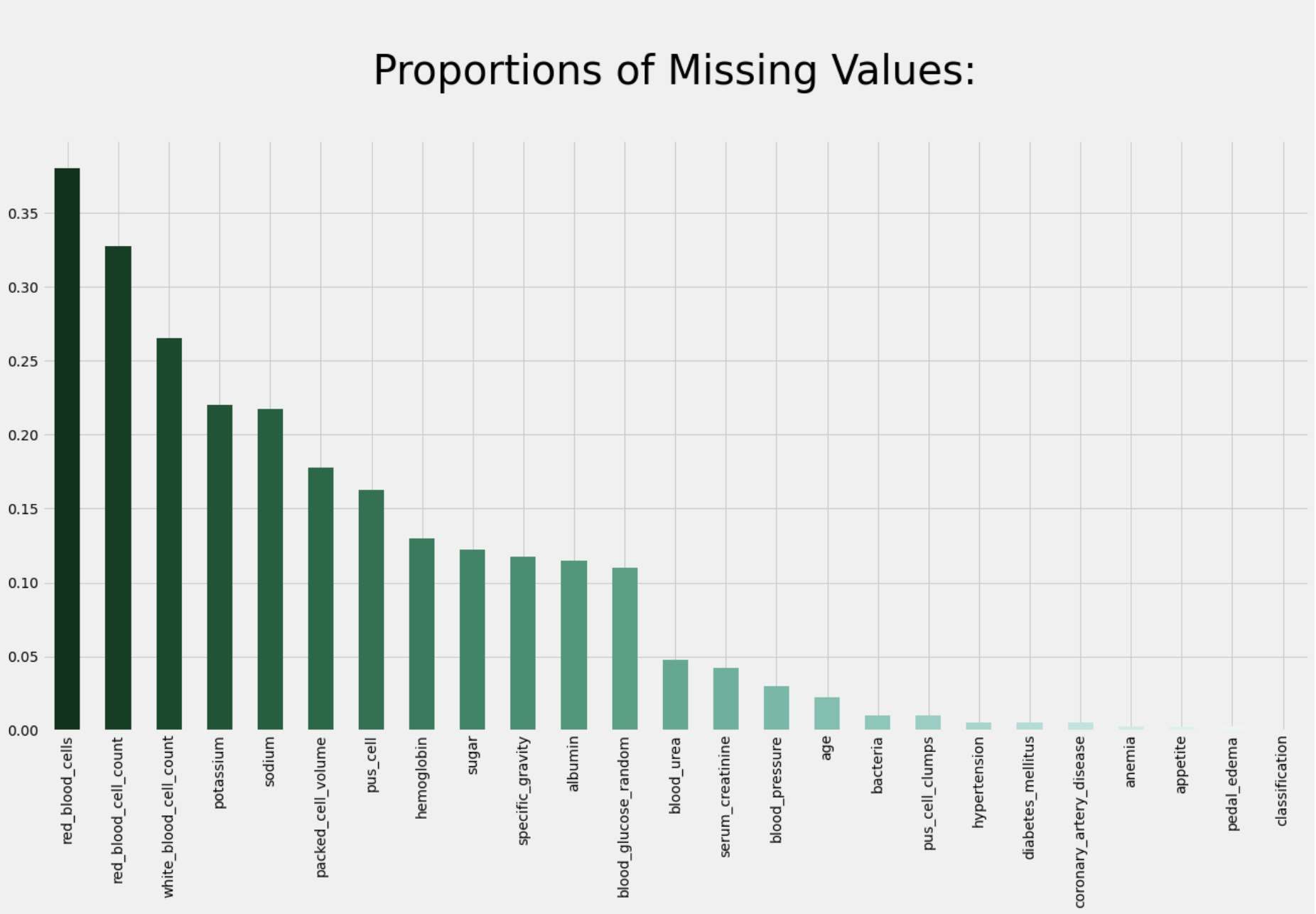
```
In [54]: for i in range(ckd_df.shape[0]):
    if ckd_df.iloc[i,24]=='ckd':
        ckd_df.iloc[i,24]='1'
    if ckd_df.iloc[i,24]=='notckd':
        ckd_df.iloc[i,24]='0'
```

### Missing Values

```
In [55]: style.use('fivethirtyeight')

d=((ckd_df.isnull().sum()/ckd_df.shape[0])).sort_values(ascending=False)
d.plot(kind='bar',
        color=sns.cubehelix_palette(start=2,
                                     rot=0.15,
                                     dark=0.15,
                                     light=0.95,
                                     reverse=True,
                                     n_colors=24),

        figsize=(20,10))
plt.title("\nProportions of Missing Values:\n",fontsize=40)
plt.show()
```



In [56]: ckd\_df.head(10).T

Out[56]:

	0	1	2	3	4	5	6	7	8	9
age	48	7	62	48	51	60	68	24	52	53
blood_pressure	80	50	80	70	80	90	70	NaN	100	90
specific_gravity	1.02	1.02	1.01	1.005	1.01	1.015	1.01	1.015	1.015	1.02
albumin	1	4	2	4	2	3	0	2	3	2
sugar	0	0	3	0	0	0	0	4	0	0
red_blood_cells	NaN	NaN	normal	normal	normal	NaN	NaN	normal	normal	abnormal
pus_cell	normal	normal	normal	abnormal	normal	NaN	normal	abnormal	abnormal	abnormal
pus_cell_clumps	notpresent	notpresent	notpresent	present	notpresent	notpresent	notpresent	notpresent	present	present
bacteria	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent	notpresent
blood_glucose_random	121	NaN	423	117	106	74	100	410	138	70
blood_urea	36	18	53	56	26	25	54	31	60	107
serum_creatinine	1.2	0.8	1.8	3.8	1.4	1.1	24	1.1	1.9	7.2
sodium	NaN	NaN	NaN	111	NaN	142	104	NaN	NaN	114
potassium	NaN	NaN	NaN	2.5	NaN	3.2	4	NaN	NaN	3.7
hemoglobin	15.4	11.3	9.6	11.2	11.6	12.2	12.4	12.4	10.8	9.5
packed_cell_volume	44	38	31	32	35	39	36	44	33	29
white_blood_cell_count	7800	6000	7500	6700	7300	7800	NaN	6900	9600	12100
red_blood_cell_count	5.2	NaN	NaN	3.9	4.6	4.4	NaN	5	4	3.7
hypertension	yes	no	no	yes	no	yes	no	no	yes	yes
diabetes_mellitus	yes	no	yes	no	no	yes	no	yes	yes	yes
coronary_artery_disease	no	no	no	no	no	no	no	no	no	no
appetite	good	good	poor	poor	good	good	good	good	good	poor
pedal_edema	no	no	no	yes	no	yes	no	yes	no	no
anemia	no	no	yes	yes	no	no	no	no	yes	yes
classification	1	1	1	1	1	1	1	1	1	1

In [58]: ckd\_df = ckd\_df.dropna()  
ckd\_df

Out[58]:

	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_cell	pus_cell_clumps	bacteria	blood_glucose_random	...	packed_cell_volume
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	29
9	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	70.0	...	29
11	63.0	70.0	1.010	3.0	0.0	abnormal	abnormal	present	notpresent	380.0	...	29
14	68.0	80.0	1.010	3.0	2.0	normal	abnormal	present	present	157.0	...	29
20	61.0	80.0	1.015	2.0	0.0	abnormal	abnormal	notpresent	notpresent	173.0	...	29
...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	140.0	...	29
396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	75.0	...	29
397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	100.0	...	29
398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	114.0	...	29
399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	131.0	...	29

158 rows × 25 columns

Hierarchical clustering algorithm is of two types:

- i) Agglomerative Hierarchical clustering algorithm or AGNES (agglomerative nesting) and
- ii) Divisive Hierarchical clustering algorithm or DIANA (divisive analysis).

Both this algorithm are exactly reverse of each other. So we will be covering Agglomerative Hierarchical clustering algorithm in detail.

Agglomerative Hierarchical clustering -This algorithm works by grouping the data one by one on the basis of the nearest distance measure of all the pairwise distance between the data point. Again distance between the data point is recalculated but which distance to consider when the groups has been formed? For this there are many available methods. Some of them are:

- 1) single-nearest distance or single linkage.

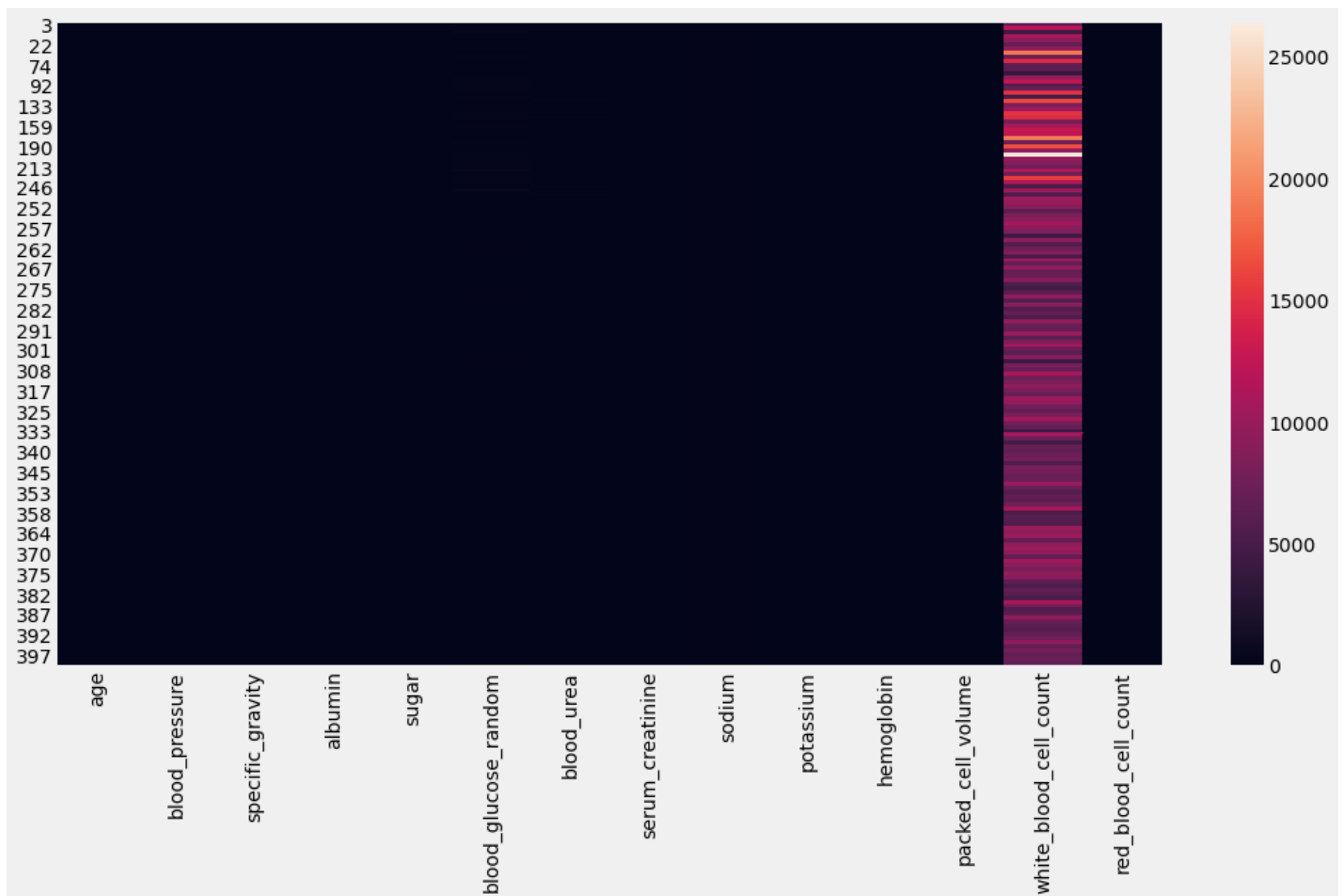
- 2) complete-farthest distance or complete linkage.
- 3) average-average distance or average linkage.
- 4) centroid distance.
- 5) ward's method - sum of squared euclidean distance is minimized.

This way we go on grouping the data until one cluster is formed. Now on the basis of dendrogram graph we can calculate how many number of clusters should be actually present.

## Heatmap

A heat map is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.

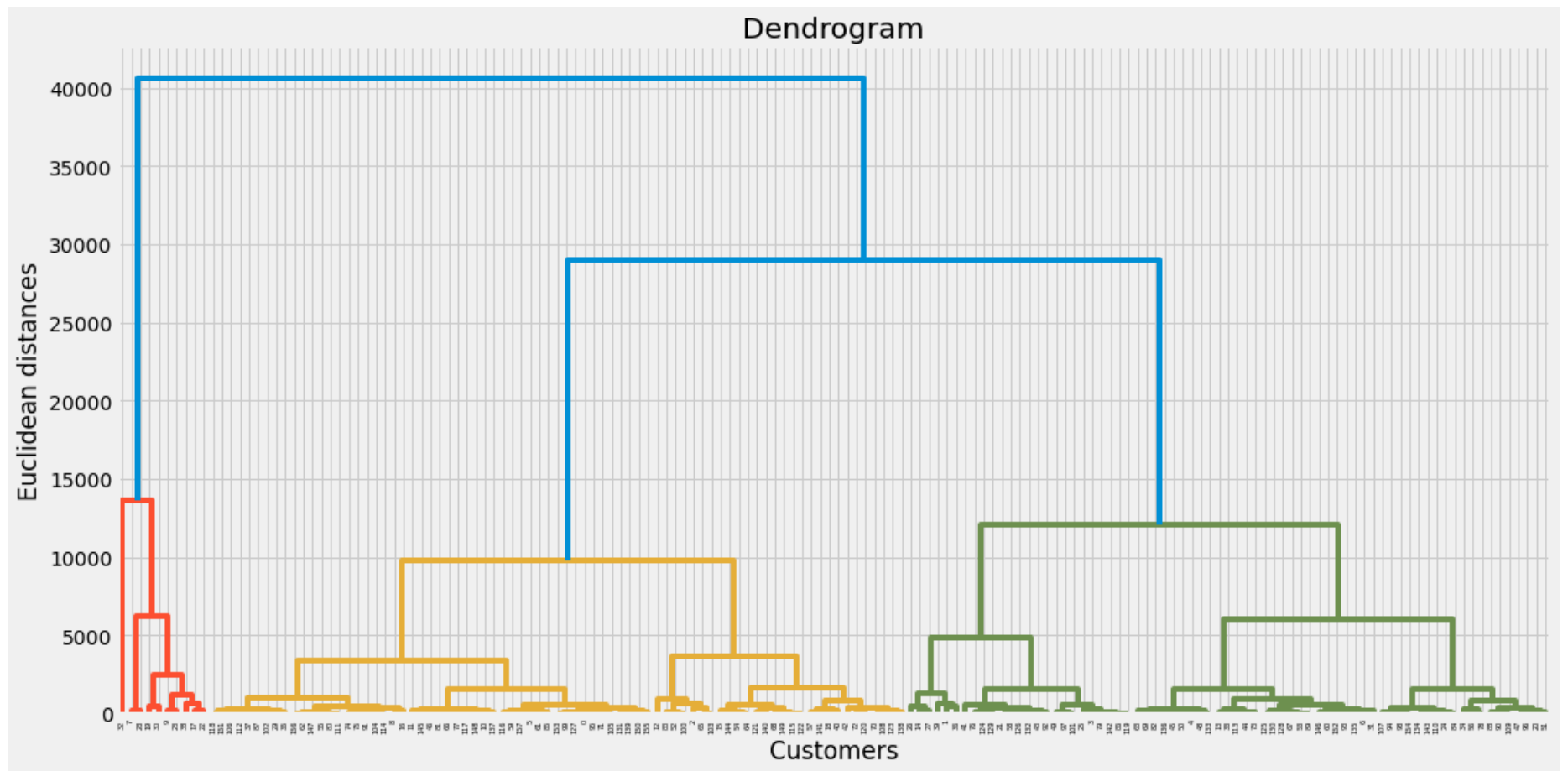
```
In [60]: plt.figure(1, figsize = (16 ,8))
sns.heatmap(ckd_df[numeric])
plt.show()
```



## Dendrogram

A dendrogram is a diagram representing a tree. This diagrammatic representation is frequently used in different contexts: in hierarchical clustering, it illustrates the arrangement of the clusters produced by the corresponding analyses.

```
In [61]: plt.figure(1, figsize = (16 ,8))
dendrogram = sch.dendrogram(sch.linkage(ckd_df[numeric], method = "ward"))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



### One-Hot Encoding



```
In [20]: onehotdata=pd.get_dummies(ckd_df,drop_first=True,prefix_sep=': ')
onehotdata.head(13).T
```

Out[20]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
age	48.00	7.00	62.00	48.000	51.00	60.000	68.00	24.000	52.000	53.00	50.00	63.00	68.000
blood_pressure	80.00	50.00	80.00	70.000	80.00	90.000	70.00	NaN	100.000	90.00	60.00	70.00	70.000
specific_gravity	1.02	1.02	1.01	1.005	1.01	1.015	1.01	1.015	1.015	1.02	1.01	1.01	1.015
albumin	1.00	4.00	2.00	4.000	2.00	3.000	0.00	2.000	3.000	2.00	2.00	3.00	3.000
sugar	0.00	0.00	3.00	0.000	0.00	0.000	0.00	4.000	0.000	0.00	4.00	0.00	1.000
blood_glucose_random	121.00	NaN	423.00	117.000	106.00	74.000	100.00	410.000	138.000	70.00	490.00	380.00	208.000
blood_urea	36.00	18.00	53.00	56.000	26.00	25.000	54.00	31.000	60.000	107.00	55.00	60.00	72.000
serum_creatinine	1.20	0.80	1.80	3.800	1.40	1.100	24.00	1.100	1.900	7.20	4.00	2.70	2.100
sodium	NaN	NaN	NaN	111.000	NaN	142.000	104.00	NaN	NaN	114.00	NaN	131.00	138.000
potassium	NaN	NaN	NaN	2.500	NaN	3.200	4.00	NaN	NaN	3.70	NaN	4.20	5.800
hemoglobin	15.40	11.30	9.60	11.200	11.60	12.200	12.40	12.400	10.800	9.50	9.40	10.80	9.700
packed_cell_volume	44.00	38.00	31.00	32.000	35.00	39.000	36.00	44.000	33.000	29.00	28.00	32.00	28.000
white_blood_cell_count	7800.00	6000.00	7500.00	6700.000	7300.00	7800.000	NaN	6900.000	9600.000	12100.00	NaN	4500.00	12200.000
red_blood_cell_count	5.20	NaN	NaN	3.900	4.60	4.400	NaN	5.000	4.000	3.70	NaN	3.80	3.400
classification	1.00	1.00	1.00	1.000	1.00	1.000	1.00	1.000	1.000	1.00	1.00	1.00	1.000
red_blood_cells: normal	0.00	0.00	1.00	1.000	1.00	0.000	0.00	1.000	1.000	0.00	0.00	0.00	0.000
pus_cell: normal	1.00	1.00	1.00	0.000	1.00	0.000	1.00	0.000	0.000	0.00	0.00	0.00	1.000
pus_cell_clumps: present	0.00	0.00	0.00	1.000	0.00	0.000	0.00	0.000	1.000	1.00	1.00	1.00	1.000
bacteria: present	0.00	0.00	0.00	0.000	0.00	0.000	0.00	0.000	0.000	0.00	0.00	0.00	0.000
hypertension: yes	1.00	0.00	0.00	1.000	0.00	1.000	0.00	0.000	1.000	1.00	1.00	1.00	1.000
diabetes_mellitus: yes	1.00	0.00	1.00	0.000	0.00	1.000	0.00	1.000	1.000	1.00	1.00	1.00	1.000
coronary_artery_disease: yes	0.00	0.00	0.00	0.000	0.00	0.000	0.00	0.000	0.000	0.00	0.00	0.00	1.000
appetite: poor	0.00	0.00	1.00	1.000	0.00	0.000	0.00	0.000	0.000	1.00	0.00	1.00	1.000
pedal_edema: yes	0.00	0.00	0.00	1.000	0.00	1.000	0.00	1.000	0.000	0.00	0.00	1.00	1.000
anemia: yes	0.00	0.00	1.00	1.000	0.00	0.000	0.00	0.000	1.000	1.00	1.00	0.00	0.000

```
In [21]: # define imputer
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean')

impute_columns=list(set(onehotdata.columns)-set(["classification"]))
print(impute_columns)

['blood_glucose_random', 'red_blood_cells: normal', 'sugar', 'potassium', 'hemoglobin', 'age', 'bacteria: present', 'anemia: yes', 'pus_cell_clumps: present', 'white_blood_cell_count', 'blood_pressure', 'serum_creatinine', 'coronary_artery_disease: yes', 'red_blood_cell_count', 'blood_urea', 'diabetes_mellitus: yes', 'albumin', 'sodium', 'appetite: poor', 'specific_gravity', 'pus_cell: normal', 'pedal_edema: yes', 'hypertension: yes', 'packed_cell_volume']
```

```
In [22]: imputer.fit(onehotdata[impute_columns])
```

Out[22]: KNNImputer()

```
In [23]: X_trans=pd.DataFrame(imputer.transform(onehotdata[impute_columns]), columns=impute_columns)
```

In [24]: X\_trans.head(13).T

Out[24]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
blood_glucose_random	121.00	113.00	423.00	117.000	106.00	74.000	100.00	410.000	138.000	70.00	490.00	380.00	208.000
red_blood_cells: normal	0.00	0.00	1.00	1.000	1.00	0.000	0.00	1.000	1.000	0.00	0.00	0.00	0.000
sugar	0.00	0.00	3.00	0.000	0.00	0.000	0.00	4.000	0.000	0.00	4.00	0.00	1.000
potassium	4.20	3.92	4.20	2.500	3.98	3.200	4.00	4.200	4.960	3.70	4.56	4.20	5.800
hemoglobin	15.40	11.30	9.60	11.200	11.60	12.200	12.40	12.400	10.800	9.50	9.40	10.80	9.700
age	48.00	7.00	62.00	48.000	51.00	60.000	68.00	24.000	52.000	53.00	50.00	63.00	68.000
bacteria: present	0.00	0.00	0.00	0.000	0.00	0.000	0.00	0.000	0.000	0.00	0.00	0.00	0.000
anemia: yes	0.00	0.00	1.00	1.000	0.00	0.000	0.00	0.000	1.000	1.00	1.00	0.00	0.000
pus_cell_clumps: present	0.00	0.00	0.00	1.000	0.00	0.000	0.00	0.000	1.000	1.00	1.00	1.00	1.000
white_blood_cell_count	7800.00	6000.00	7500.00	6700.000	7300.00	7800.000	10280.00	6900.000	9600.000	12100.00	9260.00	4500.00	12200.000
blood_pressure	80.00	50.00	80.00	70.000	80.00	90.000	70.00	74.000	100.000	90.00	60.00	70.00	70.000
serum_creatinine	1.20	0.80	1.80	3.800	1.40	1.100	24.00	1.100	1.900	7.20	4.00	2.70	2.100
coronary_artery_disease: yes	0.00	0.00	0.00	0.000	0.00	0.000	0.00	0.000	0.000	0.00	0.00	0.00	1.000
red_blood_cell_count	5.20	4.96	3.80	3.900	4.60	4.400	4.64	5.000	4.000	3.70	4.92	3.80	3.400
blood_urea	36.00	18.00	53.00	56.000	26.00	25.000	54.00	31.000	60.000	107.00	55.00	60.00	72.000
diabetes_mellitus: yes	1.00	0.00	1.00	0.000	0.00	1.000	0.00	1.000	1.000	1.00	1.00	1.00	1.000
albumin	1.00	4.00	2.00	4.000	2.00	3.000	0.00	2.000	3.000	2.00	2.00	3.00	3.000
sodium	137.60	136.80	133.80	111.000	138.40	142.000	104.00	133.800	134.000	114.00	139.00	131.00	138.000
appetite: poor	0.00	0.00	1.00	1.000	0.00	0.000	0.00	0.000	0.000	1.00	0.00	1.00	1.000
specific_gravity	1.02	1.02	1.01	1.005	1.01	1.015	1.01	1.015	1.015	1.02	1.01	1.01	1.015
pus_cell: normal	1.00	1.00	1.00	0.000	1.00	0.000	1.00	0.000	0.000	0.00	0.00	0.00	1.000
pedal_edema: yes	0.00	0.00	0.00	1.000	0.00	1.000	0.00	1.000	0.000	0.00	0.00	1.00	1.000
hypertension: yes	1.00	0.00	0.00	1.000	0.00	1.000	0.00	0.000	1.000	1.00	1.00	1.00	1.000
packed_cell_volume	44.00	38.00	31.00	32.000	35.00	39.000	36.00	44.000	33.000	29.00	28.00	32.00	28.000

In [25]: X\_trans # final dataset

Out[25]:

	blood_glucose_random	red_blood_cells: normal	sugar	potassium	hemoglobin	age	bacteria: present	anemia: yes	pus_cell_clumps: present	white_blood_cell_count	...	t
0	121.0	0.0	0.0	4.20	15.4	48.0	0.0	0.0	0.0	7800.0	...	t
1	113.0	0.0	0.0	3.92	11.3	7.0	0.0	0.0	0.0	6000.0	...	t
2	423.0	1.0	3.0	4.20	9.6	62.0	0.0	1.0	0.0	7500.0	...	t
3	117.0	1.0	0.0	2.50	11.2	48.0	0.0	1.0	1.0	6700.0	...	t
4	106.0	1.0	0.0	3.98	11.6	51.0	0.0	0.0	0.0	7300.0	...	t
...	...	...	...	...	...	...	...	...	...	...	...	t
395	140.0	1.0	0.0	4.90	15.7	55.0	0.0	0.0	0.0	6700.0	...	t
396	75.0	1.0	0.0	3.50	16.5	42.0	0.0	0.0	0.0	7800.0	...	t
397	100.0	1.0	0.0	4.40	15.8	12.0	0.0	0.0	0.0	6600.0	...	t
398	114.0	1.0	0.0	4.90	14.2	17.0	0.0	0.0	0.0	7200.0	...	t
399	131.0	1.0	0.0	3.50	15.8	58.0	0.0	0.0	0.0	6800.0	...	t

400 rows × 24 columns

Algorithmic steps for Agglomerative Hierarchical clustering

Let  $X = \{x_1, x_2, x_3, \dots, x_n\}$  be the set of data points.

- 1) Begin with the disjoint clustering having level  $L(0) = 0$  and sequence number  $m = 0$ .
- 2) Find the least distance pair of clusters in the current clustering, say pair  $(r), (s)$ , according to  $d[(r),(s)] = \min d[(i),(j)]$  where the minimum is over all pairs of clusters in the current clustering.
- 3) Increment the sequence number:  $m = m + 1$ .Merge clusters  $(r)$  and  $(s)$  into a single cluster to form the next clustering  $m$ . Set the level of this clustering to  $L(m) = d[(r),(s)]$ .

4) Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new cluster, denoted (r,s) and old cluster(k) is defined in this way:  $d[(k), (r,s)] = \min(d[(k), (r)], d[(k), (s)])$ .

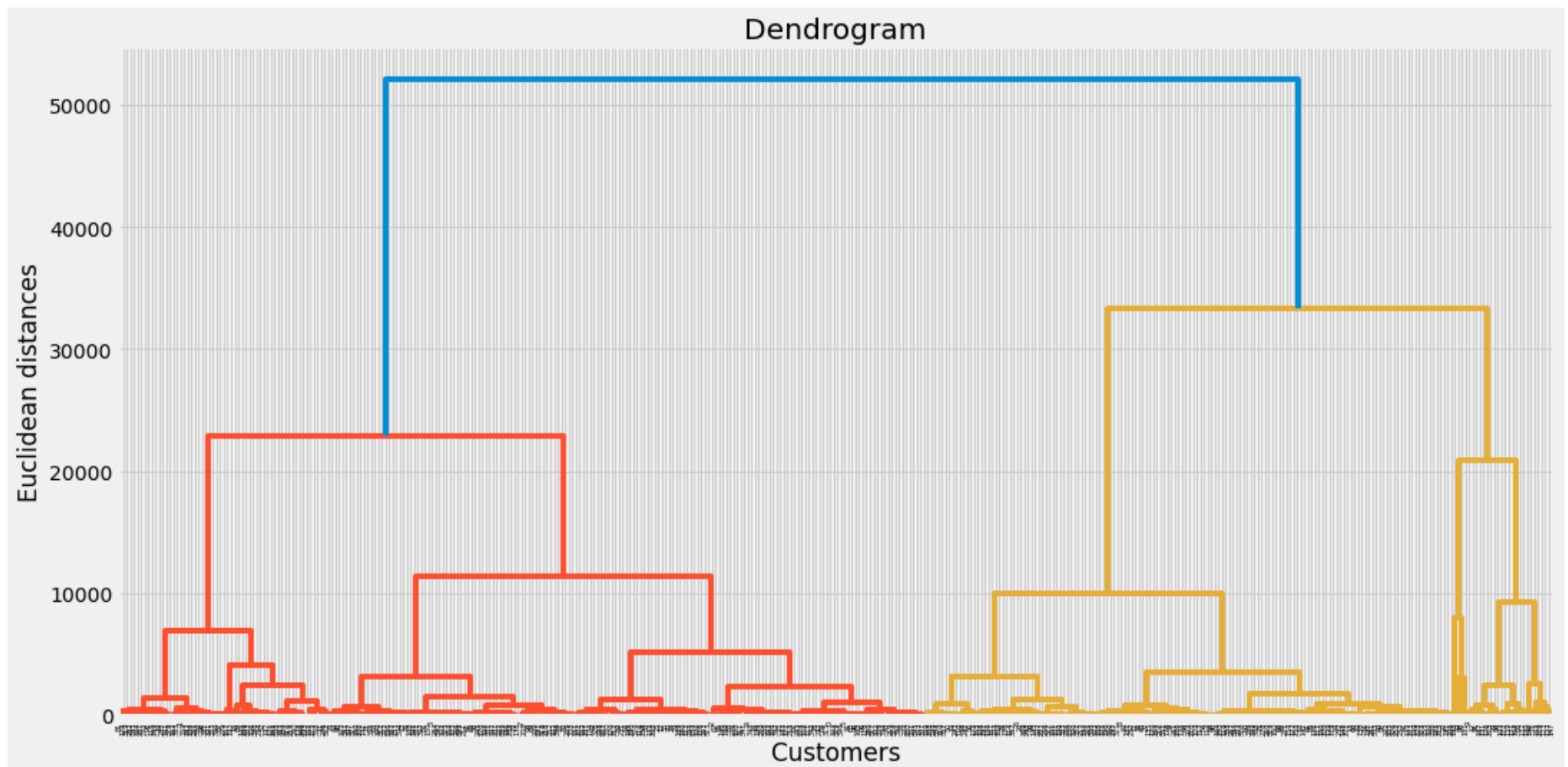
5) If all the data points are in one cluster then stop, else repeat from step 2).

Divisive Hierarchical clustering - It is just the reverse of Agglomerative Hierarchical approach.

## Dendrogram

```
In [37]: import scipy.cluster.hierarchy as sch

plt.figure(1, figsize = (16 ,8))
dendrogram = sch.dendrogram(sch.linkage(X_trans, method = "ward"))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



### Advantages

- 1) No apriori information about the number of clusters required.
- 2) Easy to implement and gives best result in some cases.

### Disadvantages

- 1) Algorithm can never undo what was done previously.
- 2) Time complexity of at least  $O(n^2 \log n)$  is required, where 'n' is the number of data points.
- 3) Based on the type of distance matrix chosen for merging different algorithms can suffer with one or more of the following:
  - i) Sensitivity to noise and outliers
  - ii) Breaking large clusters
  - iii) Difficulty handling different sized clusters and convex shapes
- 4) No objective function is directly minimized
- 5) Sometimes it is difficult to identify the correct number of clusters by the dendrogram.

In [ ]: