

Data_Preprocessing_Clean_Analysis

September 29, 2025

1 Chronic Kidney Disease dataset

Data Preprocessing & Analysis with Model Prediction Dr. Subramani Suresh

```
[176]: #Data collection
import pandas as pd
import numpy as np
import warnings
# Ignore FutureWarning
warnings.simplefilter(action='ignore', category=FutureWarning)

dataset=pd.read_csv("kidney_disease.csv")
#original dataset
dataset
```

```
[176]:
```

	id	age	bp	sg	al	su	rbc	pc	pcc	\
0	0	48.0	80.0	1.020	1.0	0.0	NaN	Normal	notpresent	
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notPresent	
2	2	62.0	80.0	1.010	2.0	3.0	Normal	normal	notpresent	
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	
..	
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	

	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	\
0	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	
1	notpresent	...	38	6000	NaN	no	no	no	good	no	no	
2	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	
3	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	
4	notpresent	...	35	7300	4.6	no	no	no	good	no	no	
..	
395	notpresent	...	47	6700	4.9	no	no	no	good	no	no	
396	notpresent	...	54	7800	6.2	no	no	no	good	no	no	
397	notpresent	...	49	6600	5.4	no	no	no	good	no	no	

```

398  notpresent  ...   51  7200  5.9   no   no   no  good  no   no
399  notpresent  ...   53  6800  6.1   no   no   no  good  no   no

```

```

      classification
0           Ckd
1           Ckd
2          ckd
3          ckd
4          ckd
..          ...
395        notckd
396        notckd
397        notckd
398        notckd
399        notckd

```

```
[400 rows x 26 columns]
```

2 Data Preprocessing

```
[177]: # columns name
dataset.dtypes
```

```

[177]: id          int64
age          float64
bp           float64
sg           float64
al           float64
su           float64
rbc          object
pc           object
pcc          object
ba           object
bgr          float64
bu           float64
sc           float64
sod          float64
pot          float64
hemo         float64
pcv          object
wc           object
rc           object
htn          object
dm           object
cad          object
appet        object

```

```

pe                object
ane               object
classification    object
dtype: object

```

```

[178]: # renames columns into more meaningful
dataset.rename(columns={
    "id": "patient_id",
    "age": "age",
    "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",
    "classification": "cdk_class"
}, inplace=True)
# after renamed
dataset.dtypes

```

```

[178]: patient_id      int64
age                  float64
blood_pressure       float64
specific_gravity     float64
albumin              float64
sugar                float64
red_blood_cells      object
pus_cell              object
pus_cell_clumps      object
bacteria              object

```

```

blood_glucose_random    float64
blood_urea              float64
serum_creatinine        float64
sodium                  float64
potassium               float64
hemoglobin              float64
packed_cell_volume      object
white_blood_cell_count  object
red_blood_cell_count    object
hypertension            object
diabetes_mellitus       object
coronary_artery_disease object
appetite                object
pedal_edema             object
anemia                  object
cdk_class               object
dtype: object

```

```

[179]: '''Descriptive statistics include those that summarize the central
tendency, dispersion and shape of a
dataset's distribution, excluding ``NaN`` values.'''
dataset.describe()

```

```

[179]:
count    patient_id      age  blood_pressure  specific_gravity  albumin  \
mean    199.500000    51.483376      76.469072      1.017408      1.016949
std     115.614301    17.169714      13.683637      0.005717      1.352679
min       0.000000      2.000000      50.000000      1.005000      0.000000
25%      99.750000     42.000000      70.000000      1.010000      0.000000
50%     199.500000     55.000000      80.000000      1.020000      0.000000
75%     299.250000     64.500000      80.000000      1.020000      2.000000
max     399.000000     90.000000     180.000000      1.025000      5.000000

```

```

count    sugar  blood_glucose_random  blood_urea  serum_creatinine  \
mean      0.450142      148.036517      57.425722      3.072454
std       1.099191      79.281714      50.503006      5.741126
min       0.000000      22.000000      1.500000      0.400000
25%       0.000000      99.000000      27.000000      0.900000
50%       0.000000     121.000000      42.000000      1.300000
75%       0.000000     163.000000      66.000000      2.800000
max        5.000000     490.000000     391.000000      76.000000

```

```

count    sodium  potassium  hemoglobin
mean    137.528754      4.627244      12.526437
std      10.408752      3.193904      2.912587

```

min	4.500000	2.500000	3.100000
25%	135.000000	3.800000	10.300000
50%	138.000000	4.400000	12.650000
75%	142.000000	4.900000	15.000000
max	163.000000	47.000000	17.800000

```
[180]: # Check each categorical (object) if uppercase exists
def df_check_uppercase(df):
    print("\nuppercase present 'yes'/'no'\n-----")
    for col in df.select_dtypes(include="object").columns:
        has_upper = df[col].str.contains(r'[A-Z]').any()
        print(f"{col}: {'uppercase present 'yes'\" if has_upper else 'no'}")
    print("\ndataset = uppercase present 'yes'\" if has_upper else '\ndataset = \u2192uppercase not present')

#check_uppercase in dataset
df_check_uppercase(dataset)
```

```
uppercase present 'yes'/'no'
-----
red_blood_cells: uppercase present 'yes'
pus_cell: uppercase present 'yes'
pus_cell_clumps: uppercase present 'yes'
bacteria: no
packed_cell_volume: no
white_blood_cell_count: no
red_blood_cell_count: no
hypertension: no
diabetes_mellitus: no
coronary_artery_disease: no
appetite: no
pedal_edema: no
anemia: no
cdk_class: uppercase present 'yes'

dataset = uppercase present 'yes'
```

```
[181]: # Check each categorical (object) column and convert all text values to \u2192lowercase
def df_convert_lowercase(df):
    for col in df.select_dtypes(include="object").columns:
        df[col] = df[col].str.lower()
    return df
#convert_lowercase in dataset
dataset=df_convert_lowercase(dataset)
#check_uppercase in dataset
```

```
df_check_uppercase(dataset)
```

```
uppercase present 'yes'/'no'
```

```
-----
```

```
red_blood_cells: no
pus_cell: no
pus_cell_clumps: no
bacteria: no
packed_cell_volume: no
white_blood_cell_count: no
red_blood_cell_count: no
hypertension: no
diabetes_mellitus: no
coronary_artery_disease: no
appetite: no
pedal_edema: no
anemia: no
cdk_class: no
```

```
dataset = uppercase not present
```

```
[182]: #Detect missing values.
dataset.isnull().sum()
```

```
[182]: patient_id          0
      age                9
      blood_pressure    12
      specific_gravity  47
      albumin           46
      sugar             49
      red_blood_cells   152
      pus_cell          65
      pus_cell_clumps    4
      bacteria          4
      blood_glucose_random 44
      blood_urea        19
      serum_creatinine  17
      sodium            87
      potassium         88
      hemoglobin        52
      packed_cell_volume 70
      white_blood_cell_count 105
      red_blood_cell_count 130
      hypertension      2
      diabetes_mellitus  2
      coronary_artery_disease 2
      appetite          1
```

```

pedal_edema          1
anemia              1
cdk_class            0
dtype: int64

```

```

[183]: #Detect missing values with order and %
def df_missing_info(df):
    missing_info = df.isnull().sum().to_frame(name='Missing')
    missing_info['% Missing (null)'] = (missing_info['Missing'] / len(df) * 100).round(2)
    missing_info = missing_info[missing_info['Missing'] > 0].
    sort_values(by='Missing', ascending=False)
    print(missing_info)

df_missing_info(dataset)

```

	Missing	% Missing (null)
red_blood_cells	152	38.00
red_blood_cell_count	130	32.50
white_blood_cell_count	105	26.25
potassium	88	22.00
sodium	87	21.75
packed_cell_volume	70	17.50
pus_cell	65	16.25
hemoglobin	52	13.00
sugar	49	12.25
specific_gravity	47	11.75
albumin	46	11.50
blood_glucose_random	44	11.00
blood_urea	19	4.75
serum_creatinine	17	4.25
blood_pressure	12	3.00
age	9	2.25
pus_cell_clumps	4	1.00
bacteria	4	1.00
hypertension	2	0.50
diabetes_mellitus	2	0.50
coronary_artery_disease	2	0.50
appetite	1	0.25
pedal_edema	1	0.25
anemia	1	0.25

```

[184]: # looking at unique values in columns
def df_uni_val_info(df):
    for col in df:
        print(f"{col} {df[col].unique()} unique values | Type :{df[col].
        dtype}\n")

```

```
df_uni_val_info(dataset)
```

```
patient_id [ 0  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] unique values | Type :int64

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 | Type :float64

blood_pressure [ 80.  50.  70.  90.  nan 100.  60. 110. 140. 180. 120.] unique
values | Type :float64

specific_gravity [1.02  1.01  1.005 1.015  nan 1.025] unique values | Type
:float64

albumin [ 1.  4.  2.  3.  0. nan  5.] unique values | Type :float64

sugar [ 0.  3.  4.  1. nan  2.  5.] unique values | Type :float64

red_blood_cells [nan 'normal' 'abnormal'] unique values | Type :object

pus_cell ['normal' 'abnormal' nan] unique values | Type :object
```


pus_cell_clumps ['notpresent' 'present' nan] unique values | Type :object

bacteria ['notpresent' 'present' nan] unique values | Type :object

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 | Type :float64

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 | Type :float64

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 | Type :float64

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 | Type :float64

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

| Type :float64

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 | Type :float64

packed_cell_volume ['44' '38' '31' '32' '35' '39' '36' '33' '29' '28' nan '16'
'24' '37' '30'

'34' '40' '45' '27' '48' '\t?' '52' '14' '22' '18' '42' '17' '46' '23'
'19' '25' '41' '26' '15' '21' '43' '20' '\t43' '47' '9' '49' '50' '53'
'51' '54'] unique values | Type :object

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' '\t6200' '7100' '11800' '9400' '5500' '5800'
'13200' '12500' '5600' '7000' '11900' '10400' '10700' '12700' '6800'
'6500' '13600' '10200' '9000' '14900' '8200' '15200' '5000' '16300'
'12400' '\t8400' '10500' '4200' '4700' '10900' '8100' '9500' '2200'
'12800' '11200' '19100' '\t?' '12300' '16700' '2600' '26400' '8800'
'7400' '4900' '8000' '12000' '15700' '4100' '5700' '11500' '5400' '10800'
'9900' '5200' '5900' '9300' '9700' '5100' '6600'] unique values | Type :object

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'
'\t?' '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 | Type :object

hypertension ['yes' 'no' nan] unique values | Type :object

diabetes_mellitus ['yes' 'no' ' yes' '\tno' '\tyes' nan] unique values | Type
:object

coronary_artery_disease ['no' 'yes' '\tno' nan] unique values | Type :object

appetite ['good' 'poor' nan] unique values | Type :object

pedal_edema ['no' 'yes' nan] unique values | Type :object

```
anemia ['no' 'yes' nan] unique values | Type :object
```

```
cdk_class ['ckd' 'ckd\t' 'notckd'] unique values | Type :object
```

```
[185]: # Clean the data based unique values, replace incorrect values eg. '\t6200',  
↳ '\t?', '\tno', '\tyes' etc  
def df_clean_all(df):  
    for col in df.columns:  
        # convert everything to string  
        df[col] = df[col].astype(str)  
        # remove tabs  
        df[col] = df[col].str.replace('\t', ' ', regex=False)  
        # Step 3: remove newlines (\n newlines, \r carriage returns)  
        df[col] = df[col].str.replace('\n', ' ', regex=False)  
        df[col] = df[col].str.replace('\r', ' ', regex=False)  
        # replace multiple spaces with a single space  
        df[col] = df[col].str.replace(' +', ' ', regex=True)  
        # strip leading and trailing spaces  
        df[col] = df[col].str.strip()  
        # replace empty or "nan" strings with real NaN  
        df[col] = df[col].replace(['', ' ', 'nan', 'NaN', 'None'], np.nan)  
        #Extra string remove  
        df[col] = df[col].replace(['?'], np.nan)  
        # try converting back to numeric if possible  
        df[col] = pd.to_numeric(df[col], errors='ignore')  
    return df  
  
dataset=df_clean_all(dataset)
```

```
[225]: # looking at unique values after clean columns  
df_uni_val_info(dataset[["diabetes_mellitus","red_blood_cell_count"]])
```

```
diabetes_mellitus ['yes' 'no'] unique values | Type :object
```

```
red_blood_cell_count [5.2  3.58 3.56 3.9  4.6  4.4  4.22 5.   4.   3.7  4.26 3.8  
3.4  3.92  
2.6  2.8  4.36 4.3  3.2  3.6  3.64 4.1  4.12 4.02 3.52 3.76 3.48 4.9  
3.82 2.5  3.84 4.2  4.16 4.5  3.1  4.7  4.24 3.5  3.42 3.74 3.62 4.06  
3.72 4.76 4.12 4.04 6.   4.46 3.98 4.62 3.94 4.52 2.1  4.44 4.28 5.6  
4.42 4.82 2.3  4.68 3.96 4.52 4.14 4.48 4.46 4.34 3.26 3.28 3.98 2.9  
3.72 2.7  3.92 8.   4.54 5.1  3.52 3.3  3.   5.3  4.54 2.4  4.8  4.86  
5.4  4.08 3.18 4.88 4.32 4.06 3.76 4.16 4.02 3.46 4.14 4.68 5.26 5.32  
4.34 5.36 4.82 6.1  5.12 6.2  6.3  5.1  5.8  5.5  5.28 6.4  5.7  5.9  
5.44 6.5  4.86 5.42 4.94] unique values | Type :float64
```

```
[187]: df_missing_info(dataset)
```

| | Missing | % Missing (null) |
|-------------------------|---------|------------------|
| red_blood_cells | 152 | 38.00 |
| red_blood_cell_count | 131 | 32.75 |
| white_blood_cell_count | 106 | 26.50 |
| potassium | 88 | 22.00 |
| sodium | 87 | 21.75 |
| packed_cell_volume | 71 | 17.75 |
| pus_cell | 65 | 16.25 |
| hemoglobin | 52 | 13.00 |
| sugar | 49 | 12.25 |
| specific_gravity | 47 | 11.75 |
| albumin | 46 | 11.50 |
| blood_glucose_random | 44 | 11.00 |
| blood_urea | 19 | 4.75 |
| serum_creatinine | 17 | 4.25 |
| blood_pressure | 12 | 3.00 |
| age | 9 | 2.25 |
| pus_cell_clumps | 4 | 1.00 |
| bacteria | 4 | 1.00 |
| hypertension | 2 | 0.50 |
| diabetes_mellitus | 2 | 0.50 |
| coronary_artery_disease | 2 | 0.50 |
| appetite | 1 | 0.25 |
| pedal_edema | 1 | 0.25 |
| anemia | 1 | 0.25 |

2.1 Missing Values

When to Remove Nulls Based on % Rules of Thumb (Industry Practice) 1. Column removal: drop columns if >70–80% missing. 2. Row removal: drop rows if >50% missing values (depends on dataset size). 3. Otherwise: use imputation (mean, median, mode, regression, KNN, etc.).

2.2 Imputation : KNNImputer

Missing values max 38%, so use imputation method: KNNImputer. The KNNImputer (from sklearn.impute) fills missing values by finding the k nearest neighbors (rows) based on other feature values. It replaces the missing value with the mean (default), median, or weighted average of those neighbors.

n_neighbors (int, default=5) # we can tune 2,3,5,6,... Number of neighboring samples to use for imputation.

2.3 Benefits of KNNImputer

1. Uses similar rows (neighbors) instead of global mean/median.
2. Preserves data patterns and variability.
3. Works for both numeric & categorical (after encoding).

4. Makes no strong assumptions about distribution.
5. Adaptive → imputes differently for different groups.
6. Often improves model accuracy compared to simple imputers.

```
[188]: # Handle missing values using the KNNImputer method
def df_fill_missing_values_knn(df):
    import pandas as pd
    import numpy as np
    from sklearn.preprocessing import LabelEncoder
    from sklearn.impute import KNNImputer
    from sklearn.model_selection import train_test_split, cross_val_score
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import accuracy_score, f1_score, roc_auc_score

    # --- STEP 1: Identify categorical and numeric columns ---
    cat_cols = df.select_dtypes(include=['object']).columns
    num_cols = df.select_dtypes(exclude=['object']).columns
    #print("Categorical columns:\n", list(cat_cols))
    #print("Numeric columns:\n", list(num_cols))

    # --- STEP 2: Encode categorical columns -----
    encoders = {}
    for col in cat_cols:
        le = LabelEncoder()
        df[col] = df[col].astype(str) # ensure string
        df[col] = df[col].replace("nan", np.nan)
        mask = df[col].notna()
        if mask.sum() > 0:
            df.loc[mask, col] = le.fit_transform(df.loc[mask, col])
            encoders[col] = le

    # --- STEP 3: Apply KNN Imputer for categorical and numeric columns----
    df = df.apply(pd.to_numeric, errors='coerce')
    imputer = KNNImputer(n_neighbors=5)
    df_imputed = imputer.fit_transform(df)
    df_imputed = pd.DataFrame(df_imputed, columns=df.columns)

    #Encoded and imputed dataset (dataset contain numeric value only) for any
    ↪ model creation
    df_imputed_encoded = df_imputed.copy()

    #--- STEP 4: Valiation with KNeighborsClassifier Model-----

    #target columns for prediction
    # get the last column name as target
    target_col = df.columns[-1]
    # X, y split
```

```

X=df_imputed.drop(target_col, axis=1)
y=df_imputed[target_col]

#split train and test set 20% test data 80% train data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪20,random_state=42)

#model creation
model=KNeighborsClassifier(n_neighbors=5) #5
model.fit(X_train,y_train)

#y prediction for valitation
y_pred = model.predict(X_test)

# Valitation result
print(f"\nImputation Validation: Based on KNN model\nTrain Set:")
# Training set evaluation
print(f"Model accuracy: {model.score(X_train,y_train):.2f}")
# Test set evaluation
print(f"Test Set:")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print(f"F1_score: {f1_score(y_test, y_pred):.2f}")
print(f"ROC-AUC : {roc_auc_score(y_test, y_pred):.2f}")
# Cross-validation
cv_scores = cross_val_score(model, X, y, cv=10) # 10 cross-fold CV
print("Model Cross-Validation (10-fold) score:", cv_scores.mean())

# --- STEP 4: Decode categorical columns -----
# Decode for data analysis
for col in cat_cols:
    if col in encoders:
        df_imputed[col] = df_imputed[col].round().astype(int)
        df_imputed[col] = encoders[col].inverse_transform(df_imputed[col])

# df_imputed_encoded = dataset after encoding all categorical and numerical
↪columns
# df_imputed = dataset after decoding categorical columns back to original
↪labels
    return df_imputed, df_imputed_encoded
#
#call function
dataset,dataset_encoded=df_fill_missing_values_knn(dataset)

```

Imputation Validation: Based on KNN model
 Train Set:
 Model accuracy: 0.96

Test Set:
 Accuracy: 0.97
 F1_score: 0.97
 ROC-AUC : 0.98
 Model Cross-Validation (10-fold) score: 0.9

```
[226]: # dataset after decoding categorical columns back to original labels
dataset
```

```
[226]:      patient_id  age  blood_pressure  specific_gravity  albumin  sugar  \
0           0.0  48.0           80.0           1.020        1.0    0.0
1           1.0   7.0           50.0           1.020        4.0    0.0
2           2.0  62.0           80.0           1.010        2.0    3.0
3           3.0  48.0           70.0           1.005        4.0    0.0
4           4.0  51.0           80.0           1.010        2.0    0.0
..          ...  ...          ...          ...          ...    ...
395        395.0  55.0           80.0           1.020        0.0    0.0
396        396.0  42.0           70.0           1.025        0.0    0.0
397        397.0  12.0           80.0           1.020        0.0    0.0
398        398.0  17.0           60.0           1.025        0.0    0.0
399        399.0  58.0           80.0           1.025        0.0    0.0
```

```
      red_blood_cells  pus_cell  pus_cell_clumps  bacteria  ...  \
0          abnormal    normal    notpresent  notpresent  ...
1          abnormal    normal    notpresent  notpresent  ...
2           normal    normal    notpresent  notpresent  ...
3           normal  abnormal      present  notpresent  ...
4           normal    normal    notpresent  notpresent  ...
..           ...      ...          ...      ...
395          normal    normal    notpresent  notpresent  ...
396          normal    normal    notpresent  notpresent  ...
397          normal    normal    notpresent  notpresent  ...
398          normal    normal    notpresent  notpresent  ...
399          normal    normal    notpresent  notpresent  ...
```

```
      packed_cell_volume  white_blood_cell_count  red_blood_cell_count  \
0              44.0              7800.0              5.20
1              38.0              6000.0              3.58
2              31.0              7500.0              3.56
3              32.0              6700.0              3.90
4              35.0              7300.0              4.60
..              ...              ...              ...
395             47.0             6700.0              4.90
396             54.0             7800.0              6.20
397             49.0             6600.0              5.40
398             51.0             7200.0              5.90
399             53.0             6800.0              6.10
```

| | hypertension | diabetes_mellitus | coronary_artery_disease | appetite | \ |
|-----|--------------|-------------------|-------------------------|----------|---|
| 0 | yes | yes | no | good | |
| 1 | no | no | no | good | |
| 2 | no | yes | no | poor | |
| 3 | yes | no | no | poor | |
| 4 | no | no | no | good | |
| .. | ... | ... | ... | ... | |
| 395 | no | no | no | good | |
| 396 | no | no | no | good | |
| 397 | no | no | no | good | |
| 398 | no | no | no | good | |
| 399 | no | no | no | good | |

| | pedal_edema | anemia | cdk_class |
|-----|-------------|--------|-----------|
| 0 | no | no | ckd |
| 1 | no | no | ckd |
| 2 | no | yes | ckd |
| 3 | yes | yes | ckd |
| 4 | no | no | ckd |
| .. | ... | ... | ... |
| 395 | no | no | notckd |
| 396 | no | no | notckd |
| 397 | no | no | notckd |
| 398 | no | no | notckd |
| 399 | no | no | notckd |

[400 rows x 26 columns]

```
[191]: # cleaned dataset without null/missing values
df_missing_info(dataset)
```

```
Empty DataFrame
Columns: [Missing, % Missing (null)]
Index: []
```

3 Data Analysis

```
[227]: #split categorical column names
cat_cals=[cal for cal in dataset if dataset[cal].dtype==object]
cat_cals
```

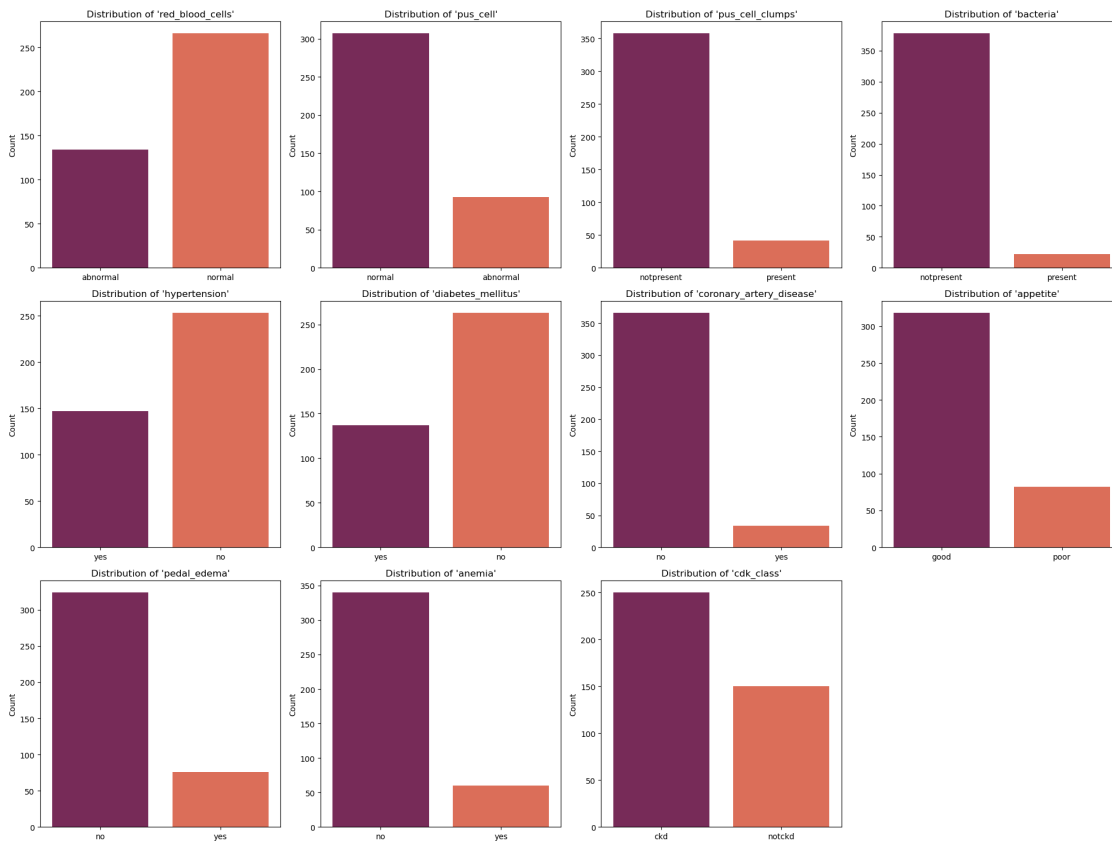
```
[227]: ['red_blood_cells',
'pus_cell',
'pus_cell_clumps',
'bacteria',
'hypertension',
```



```
'diabetes_mellitus',  
'coronary_artery_disease',  
'appetite',  
'pedal_edema',  
'anemia',  
'cdk_class']
```

4 categorical columns

```
[209]: #looking at categorical columns  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
plt.figure(figsize=(20, 15))  
plotnumber = 1  
  
for column in cat_cols:  
    if plotnumber <= 11:  
        ax = plt.subplot(3, 4, plotnumber)  
        sns.countplot(x=dataset[column], palette='rocket', ax=ax)  
        ax.set_title(f"Distribution of '{column}'") # Title above each plot  
        ax.set_xlabel("") # optional: remove x-axis label if title is enough  
        ax.set_ylabel("Count")  
  
        plotnumber += 1  
  
plt.tight_layout()  
plt.show()
```



5 Numeric column

```
[216]: #split numeric column names
num_cals=[cal for cal in dataset if dataset[cal].dtype!=object]
num_cals.remove("patient_id")
num_cals
```

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

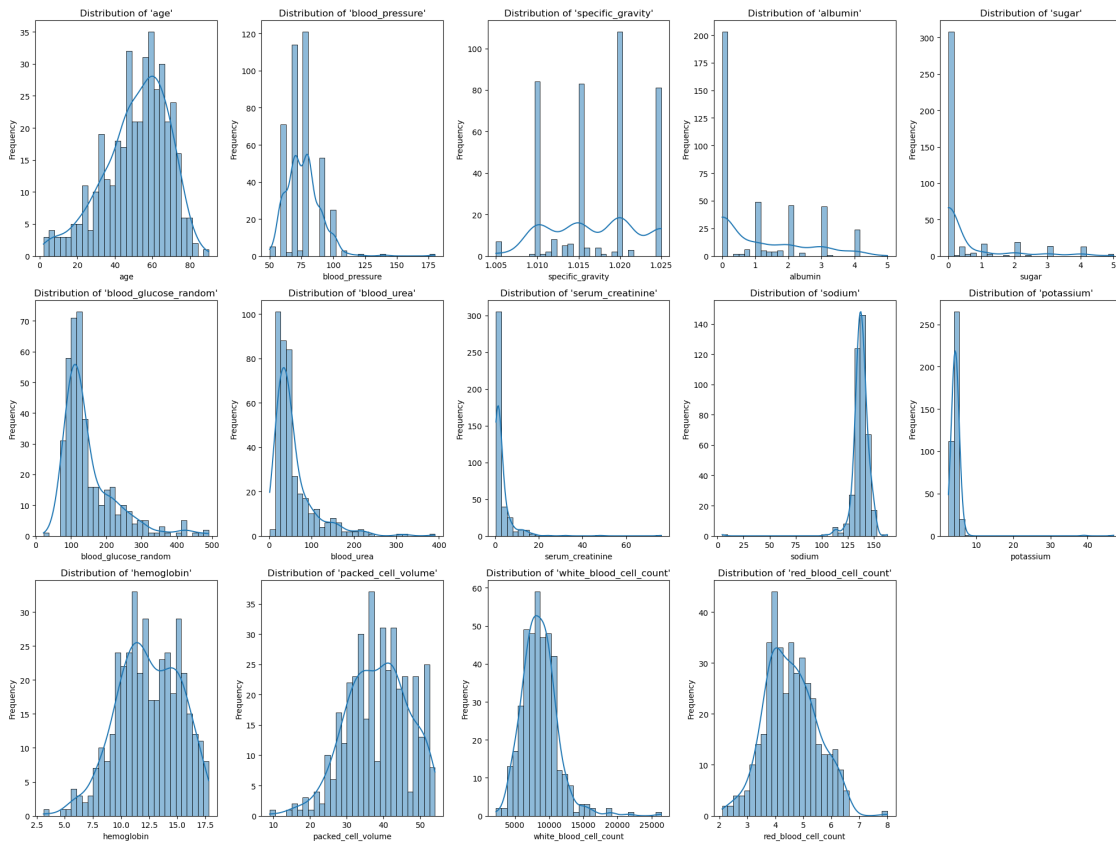
```
'red_blood_cell_count']
```

```
[219]: # checking numerical features distribution
plt.figure(figsize=(20, 15))
plotnumber = 1

for column in num_cals:
    if plotnumber <= 14: # adjust subplot grid if you have more/less features
        ax = plt.subplot(3, 5, plotnumber)
        sns.histplot(dataset[column], kde=True, bins=30) # kde=True adds
        ↪ density curve
        ax.set_title(f"Distribution of '{column}'") # Title above each plot
        plt.xlabel(column)
        plt.ylabel("Frequency")

        plotnumber += 1

plt.tight_layout()
plt.show()
```



```
[221]: #looking at numeric column

#create New Table for central Tendency & descriptive
def UnivariateTable(df,num_cals):
    import pandas as pd
    import numpy as np
    descriptive=pd.DataFrame(index=["Mean","Median","Mode",
                                   "Min","Q1-25%","Q2-50%","Q3-75%","Max",
                                   "IQR","1.5-Rule","Lower-Bound","Upper-Bound"]
                             ,columns=num_cals)

    for cal in num_cals:
        #print(cal)
        descriptive.loc["Mean", cal] = round(df[cal].mean(), 2)
        descriptive.loc["Median", cal] = df[cal].median()
        descriptive.loc["Mode", cal] = round(df[cal].mode()[0], 2)
        descriptive.loc["Min",cal]=df.describe().loc['min',cal]
        descriptive.loc["Q1-25%",cal]=df.describe().loc['25%',cal]
        descriptive.loc["Q2-50%",cal]=df.describe().loc['50%',cal]
        descriptive.loc["Q3-75%",cal]=df.describe().loc['75%',cal]
        descriptive.loc["Max",cal]=df.describe().loc['max',cal]
        descriptive.loc["IQR",cal]=descriptive.loc["Q3-75%",cal]-descriptive.
        ↪loc["Q1-25%",cal]
        descriptive.loc["1.5-Rule",cal]=1.5*descriptive.loc["IQR",cal]
        descriptive.loc["Lower-Bound",cal]=descriptive.
        ↪loc["Q1-25%",cal]-descriptive.loc["1.5-Rule",cal]
        descriptive.loc["Upper-Bound",cal]=descriptive.
        ↪loc["Q3-75%",cal]+descriptive.loc["1.5-Rule",cal]
    return descriptive

UnivariateTable(dataset,num_cals)
```

```
[221]:
```

| | age | blood_pressure | specific_gravity | albumin | sugar | \ |
|-------------|-------|----------------|------------------|---------|-------|---|
| Mean | 51.38 | 76.36 | 1.02 | 1.05 | 0.46 | |
| Median | 54.0 | 80.0 | 1.016 | 0.0 | 0.0 | |
| Mode | 60.0 | 80.0 | 1.02 | 0.0 | 0.0 | |
| Min | 2.0 | 50.0 | 1.005 | 0.0 | 0.0 | |
| Q1-25% | 42.0 | 70.0 | 1.012 | 0.0 | 0.0 | |
| Q2-50% | 54.0 | 80.0 | 1.016 | 0.0 | 0.0 | |
| Q3-75% | 64.0 | 80.0 | 1.02 | 2.0 | 0.0 | |
| Max | 90.0 | 180.0 | 1.025 | 5.0 | 5.0 | |
| IQR | 22.0 | 10.0 | 0.008 | 2.0 | 0.0 | |
| 1.5-Rule | 33.0 | 15.0 | 0.012 | 3.0 | 0.0 | |
| Lower-Bound | 9.0 | 55.0 | 1.0 | -3.0 | 0.0 | |
| Upper-Bound | 97.0 | 95.0 | 1.032 | 5.0 | 0.0 | |

| | blood_glucose_random | blood_urea | serum_creatinine | sodium | \ |
|------|----------------------|------------|------------------|--------|---|
| Mean | 150.34 | 57.15 | 3.06 | 137.24 | |

| | | | | |
|-------------|-------|-------|-------|-------|
| Median | 123.0 | 42.0 | 1.3 | 137.3 |
| Mode | 99.0 | 46.0 | 1.2 | 135.0 |
| Min | 22.0 | 1.5 | 0.4 | 4.5 |
| Q1-25% | 101.0 | 27.0 | 0.9 | 135.0 |
| Q2-50% | 123.0 | 42.0 | 1.3 | 137.3 |
| Q3-75% | 172.0 | 66.0 | 2.8 | 141.0 |
| Max | 490.0 | 391.0 | 76.0 | 163.0 |
| IQR | 71.0 | 39.0 | 1.9 | 6.0 |
| 1.5-Rule | 106.5 | 58.5 | 2.85 | 9.0 |
| Lower-Bound | -5.5 | -31.5 | -1.95 | 126.0 |
| Upper-Bound | 278.5 | 124.5 | 5.65 | 150.0 |

| | potassium | hemoglobin | packed_cell_volume | white_blood_cell_count | \ |
|-------------|-----------|------------|--------------------|------------------------|---|
| Mean | 4.56 | 12.44 | 38.35 | 8584.45 | |
| Median | 4.34 | 12.3 | 39.0 | 8300.0 | |
| Mode | 3.5 | 15.0 | 41.0 | 9800.0 | |
| Min | 2.5 | 3.1 | 9.0 | 2200.0 | |
| Q1-25% | 3.9 | 10.515 | 32.4 | 6900.0 | |
| Q2-50% | 4.34 | 12.3 | 39.0 | 8300.0 | |
| Q3-75% | 4.9 | 14.8 | 44.0 | 9800.0 | |
| Max | 47.0 | 17.8 | 54.0 | 26400.0 | |
| IQR | 1.0 | 4.285 | 11.6 | 2900.0 | |
| 1.5-Rule | 1.5 | 6.4275 | 17.4 | 4350.0 | |
| Lower-Bound | 2.4 | 4.0875 | 15.0 | 2550.0 | |
| Upper-Bound | 6.4 | 21.2275 | 61.4 | 14150.0 | |

| | red_blood_cell_count |
|-------------|----------------------|
| Mean | 4.54 |
| Median | 4.5 |
| Mode | 5.2 |
| Min | 2.1 |
| Q1-25% | 3.9 |
| Q2-50% | 4.5 |
| Q3-75% | 5.2 |
| Max | 8.0 |
| IQR | 1.3 |
| 1.5-Rule | 1.95 |
| Lower-Bound | 1.95 |
| Upper-Bound | 7.15 |

6 Model Prediction

```
[228]: #dataset after encoding all categorical and numerical columns
dataset_encoded.head()
```

```
[228]:
```

| | patient_id | age | blood_pressure | specific_gravity | albumin | sugar | \ |
|---|------------|------|----------------|------------------|---------|-------|---|
| 0 | 0.0 | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | |
| 1 | 1.0 | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | |
| 2 | 2.0 | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | |
| 3 | 3.0 | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | |
| 4 | 4.0 | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | |

| | red_blood_cells | pus_cell | pus_cell_clumps | bacteria | ... | \ |
|---|-----------------|----------|-----------------|----------|-----|---|
| 0 | 0.2 | 1.0 | 0.0 | 0.0 | ... | |
| 1 | 0.4 | 1.0 | 0.0 | 0.0 | ... | |
| 2 | 1.0 | 1.0 | 0.0 | 0.0 | ... | |
| 3 | 1.0 | 0.0 | 1.0 | 0.0 | ... | |
| 4 | 1.0 | 1.0 | 0.0 | 0.0 | ... | |

| | packed_cell_volume | white_blood_cell_count | red_blood_cell_count | \ |
|---|--------------------|------------------------|----------------------|---|
| 0 | 44.0 | 7800.0 | 5.20 | |
| 1 | 38.0 | 6000.0 | 3.58 | |
| 2 | 31.0 | 7500.0 | 3.56 | |
| 3 | 32.0 | 6700.0 | 3.90 | |
| 4 | 35.0 | 7300.0 | 4.60 | |

| | hypertension | diabetes_mellitus | coronary_artery_disease | appetite | \ |
|---|--------------|-------------------|-------------------------|----------|---|
| 0 | 1.0 | 1.0 | 0.0 | 0.0 | |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 2 | 0.0 | 1.0 | 0.0 | 1.0 | |
| 3 | 1.0 | 0.0 | 0.0 | 1.0 | |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | |

| | pedal_edema | anemia | cdk_class |
|---|-------------|--------|-----------|
| 0 | 0.0 | 0.0 | 0.0 |
| 1 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 1.0 | 0.0 |
| 3 | 1.0 | 1.0 | 0.0 |
| 4 | 0.0 | 0.0 | 0.0 |

[5 rows x 26 columns]

```
[222]: # Model prediction for encoded dataset

#split X, y
target_col="cdk_class"
X=dataset_encoded.drop(target_col, axis=1)
y=dataset_encoded[target_col]

#split train and test
from sklearn.model_selection import train_test_split
```

```

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↳30,random_state=42, shuffle=True)

#model creation
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score

models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "XGBoost": xgb.XGBClassifier()
}

# Evaluate models
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name} Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print(f"{name} f1_score: {f1_score(y_test, y_pred):.2f}")
    print(f"{name} ROC-AUC : {roc_auc_score(y_test, y_pred):.2f}")

```

C:\Anaconda3\envs\ssai\Lib\site-packages\sklearn\linear_model_logistic.py:473:
ConvergenceWarning: lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).

You might also want to scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

Logistic Regression Accuracy: 1.00

Logistic Regression f1_score: 1.00

Logistic Regression ROC-AUC : 1.00

Decision Tree Accuracy: 0.99

Decision Tree f1_score: 0.99

Decision Tree ROC-AUC : 0.99

Random Forest Accuracy: 1.00

Random Forest f1_score: 1.00

Random Forest ROC-AUC : 1.00

XGBoost Accuracy: 0.99
XGBoost f1_score: 0.99
XGBoost ROC-AUC : 0.99

7 Save Cleaned Dataset

```
[223]: # -----Save cleaned dataset -----  
dataset.to_csv("ckd_dataset_imputed.csv", index=False)  
dataset_encoded.to_csv("cdk_encoded_dataset_imputed.csv", index=False)  
print("Cleaned dataset saved. Shape:", dataset.shape, dataset_encoded.shape)
```

Cleaned dataset saved. Shape: (400, 26) (400, 26)

```
[174]: # Dr. Subramani Suresh
```

```
[ ]:
```