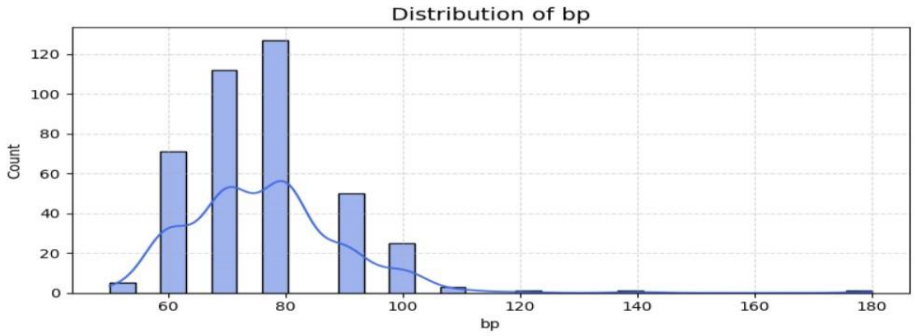
Data Collection and Preprocessing Phase

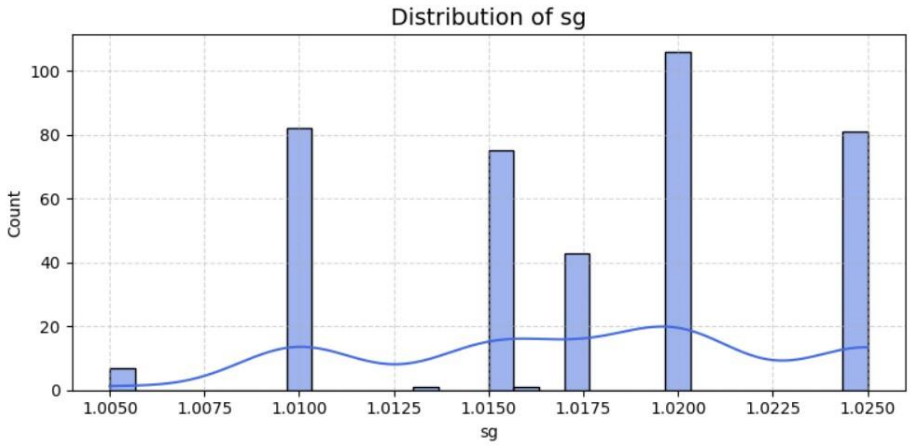
Date	16 June 2025
Team ID	SWTID1749713922
Project Title	Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management
Maximum Marks	6 Marks

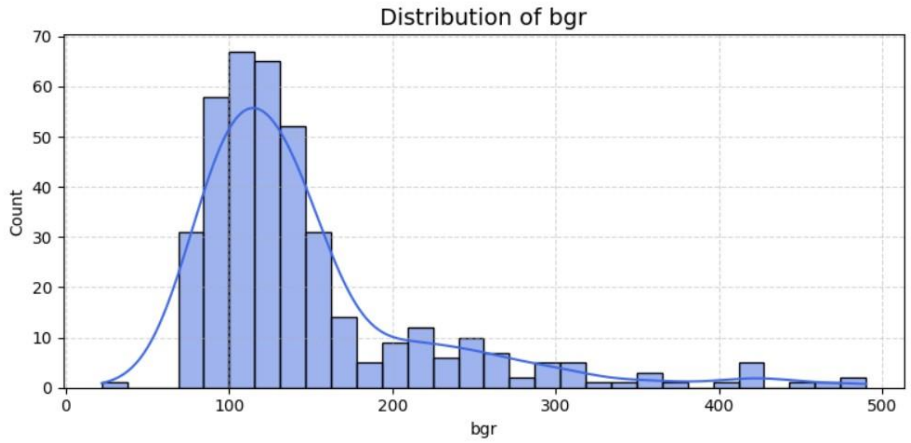
Data Exploration and Preprocessing Template

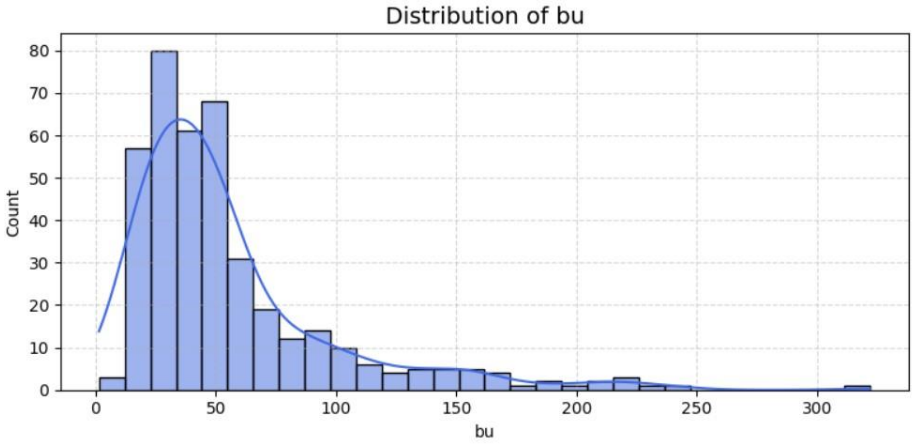
All data sources were identified and assessed for quality issues such as missing values and duplicates. Appropriate resolution measures were implemented to ensure the accuracy and reliability of the analysis.

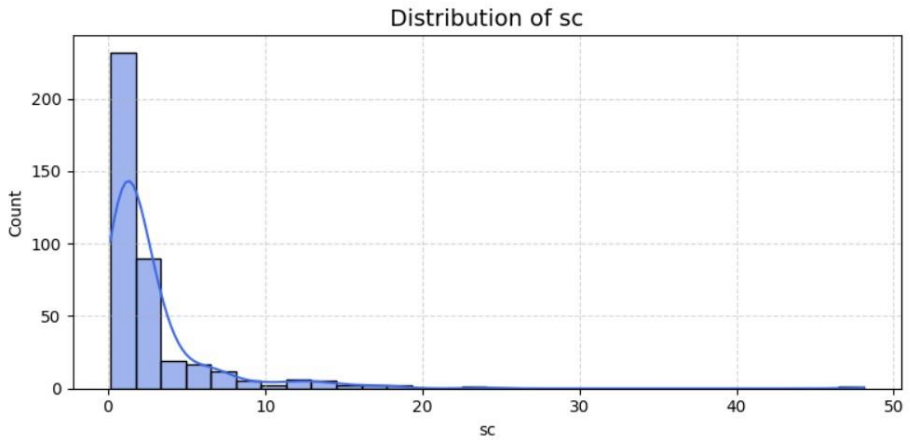
Section	Description
Data Overview	Dimensions: 400 X 26 Descriptive statistics: `df.describe()` id age bp sg al su bgr bu sc sod pot hemo count 400.000000 391.000000 388.000000 353.000000 354.000000 351.000000 356.000000 381.000000 383.000000 313.000000 312.000000 348.000000 mean 199.500000 51.483376 76.469072 1.017408 1.016949 0.450142 148.036517 57.425722 3.072454 137.528754 4.627244 12.526437 std 115.614301 17.169714 13.683637 0.005717 1.352679 1.099191 79.281714 50.503006 5.741126 10.408752 3.193904 2.912587 min 0.000000 2.000000 50.000000 1.005000 0.000000 0.000000 22.000000 1.500000 0.400000 4.500000 2.500000 3.100000 25% 99.750000 42.000000 70.000000 1.010000 0.000000 0.000000 99.000000 27.000000 0.900000 135.000000 3.800000 10.300000 50% 199.500000 55.000000 80.000000 1.020000 0.000000 0.000000 121.000000 42.000000 1.300000 138.000000 4.400000 12.650000 75% 299.250000 64.500000 80.000000 1.020000 2.000000 0.000000 163.000000 66.000000 2.800000 142.000000 4.900000 15.000000 max 399.000000 90.000000 180.000000 1.025000 5.000000 5.000000 490.000000 391.000000 76.000000 163.000000 47.000000 17.800000

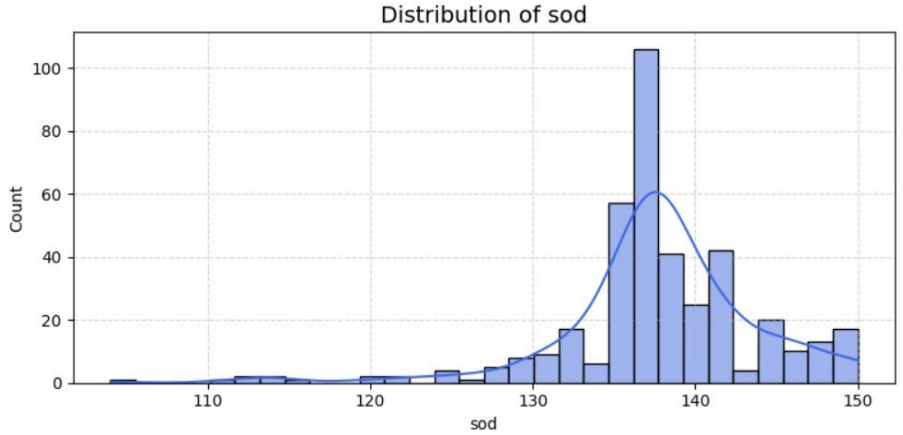
Univariate Analysis	 Distribution Plots for Numerical Variables:

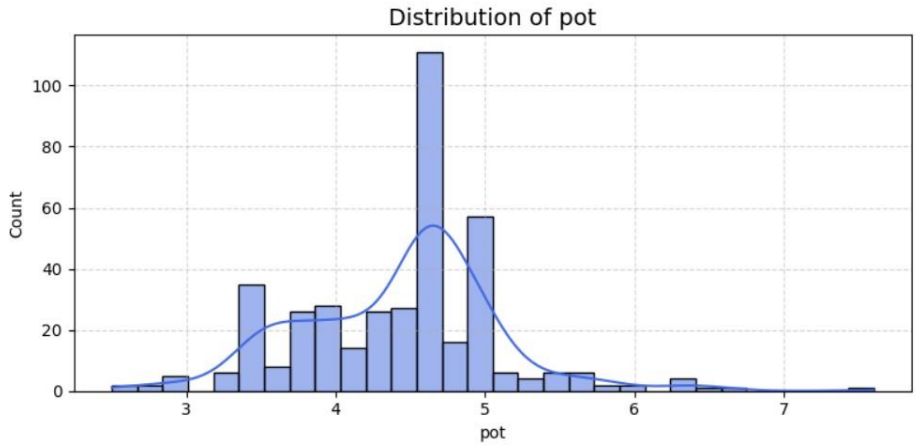

Distribution of sg
Count
sg
0
20
40
60
80
100
1.0050
1.0075
1.0100
1.0125
1.0150
1.0175
1.0200
1.0225
1.0250

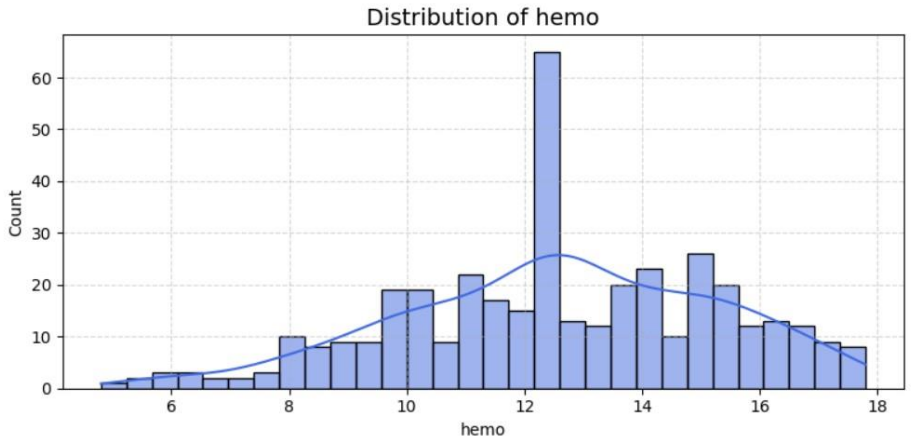

Distribution of bgr
Count
bgr
0
10
20
30
40
50
60
70
0
100
200
300
400
500

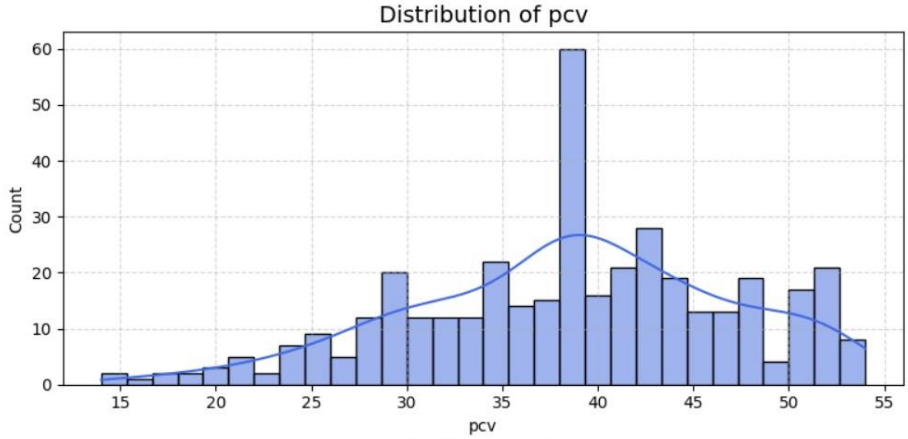

Distribution of bu
Count
bu
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300

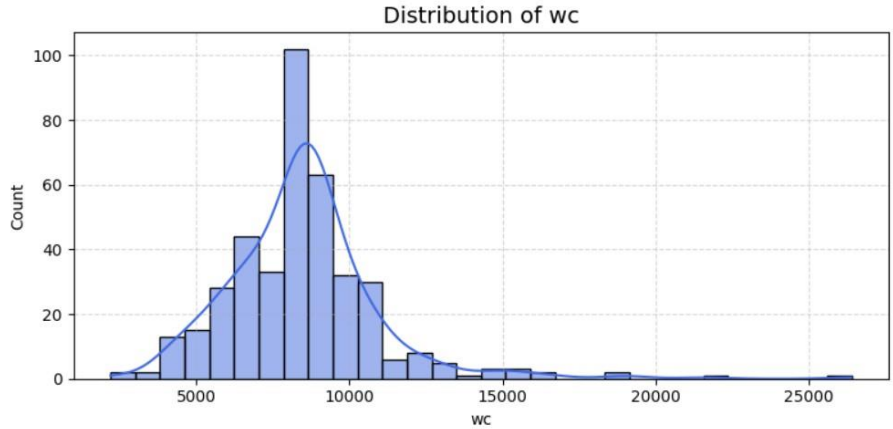

Distribution of sc
Count
sc
0
50
100
150
200
0
10
20
30
40
50

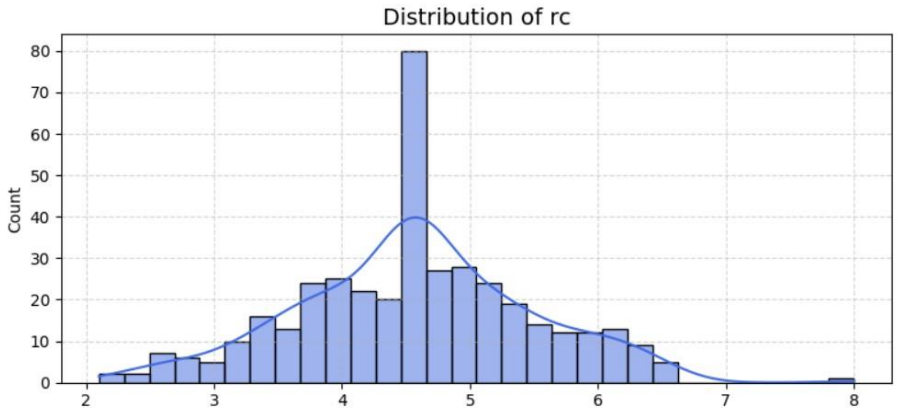

Distribution of sod
Count
sod
0
20
40
60
80
100
110
120
130
140
150

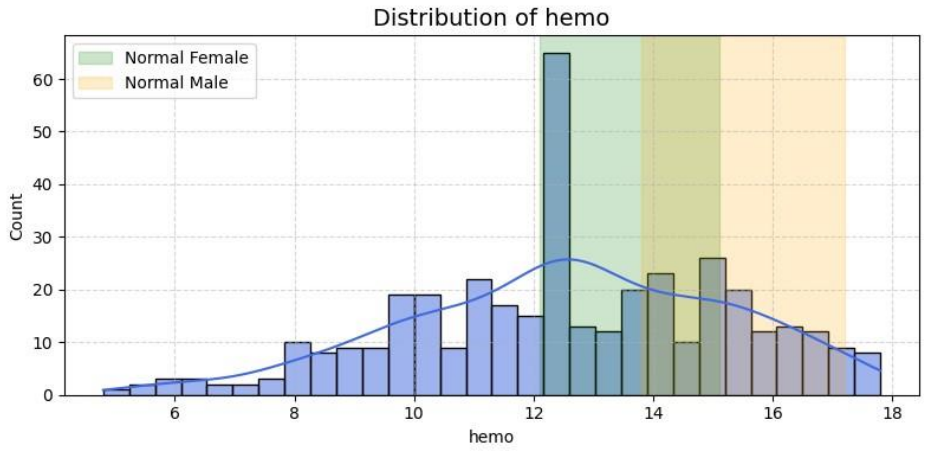

Distribution of pot
Count
pot
0
20
40
60
80
100
3
4
5
6
7

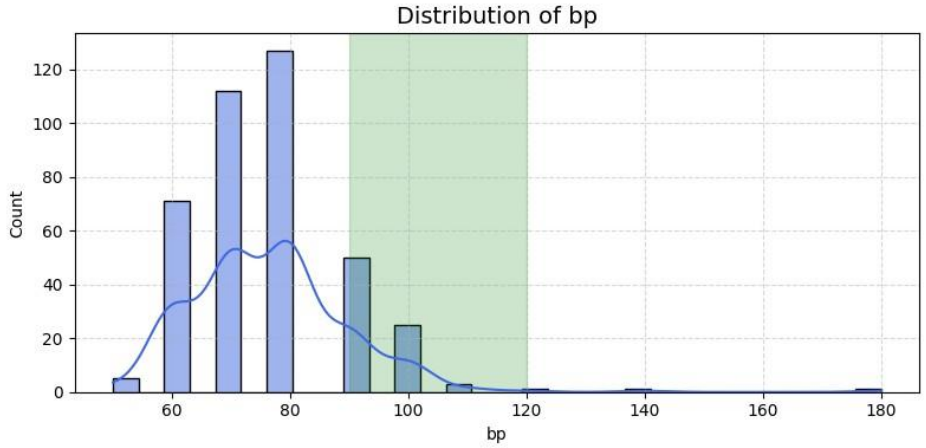

Distribution of hemo
Count
hemo
0
10
20
30
40
50
60
6
8
10
12
14
16
18


Distribution of pcv
Count
pcv
0
10
20
30
40
50
60
15
20
25
30
35
40
45
50
55

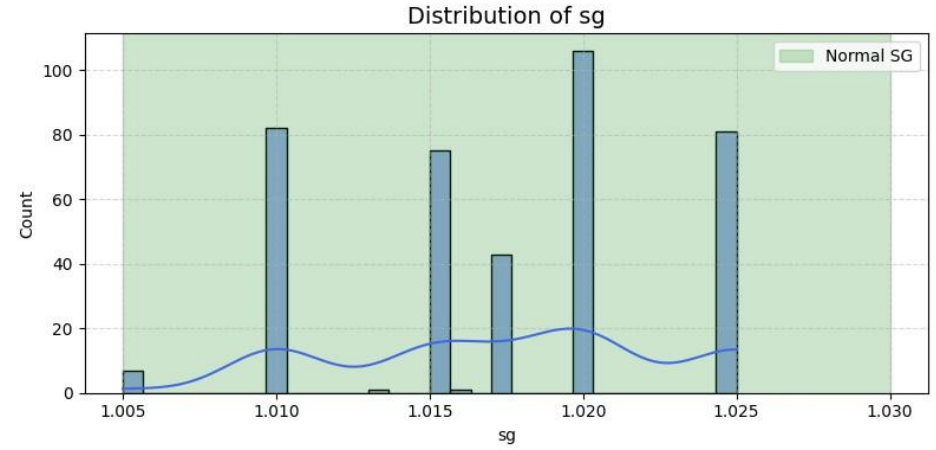

Distribution of wc
Count
wc
0
20
40
60
80
100
5000
10000
15000
20000
25000

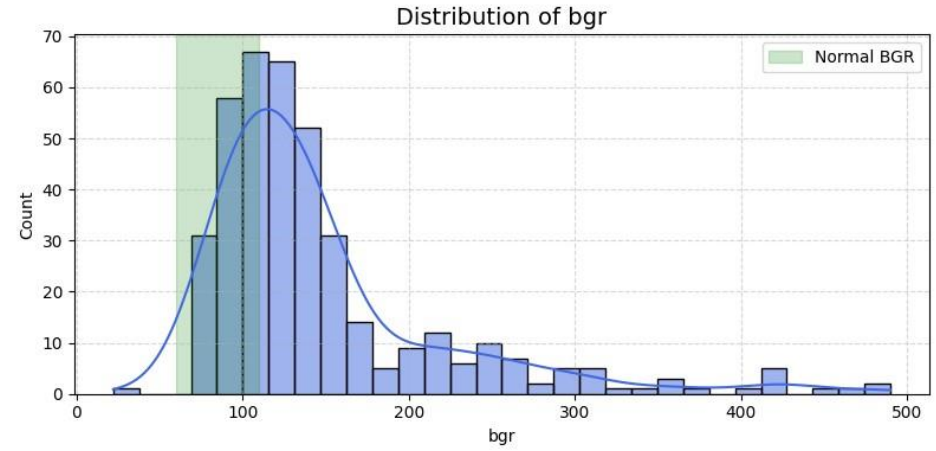

Distribution of rc
Count
0
10
20
30
40
50
60
70
80
2
3
4
5
6
7
8

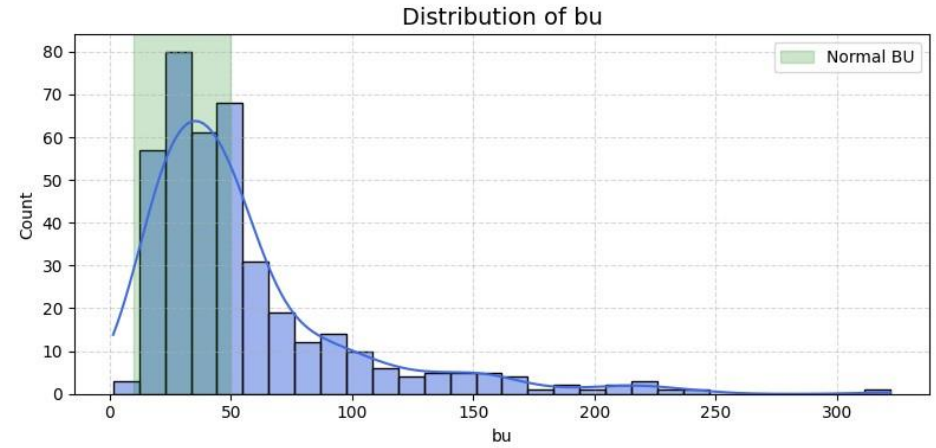


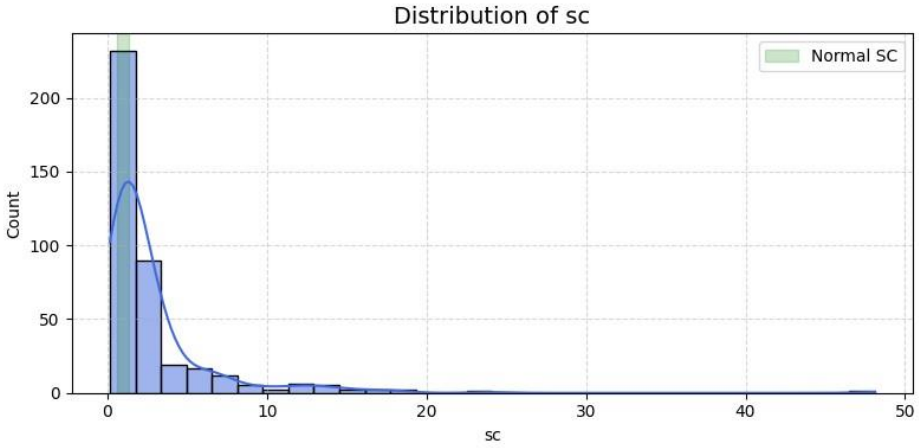


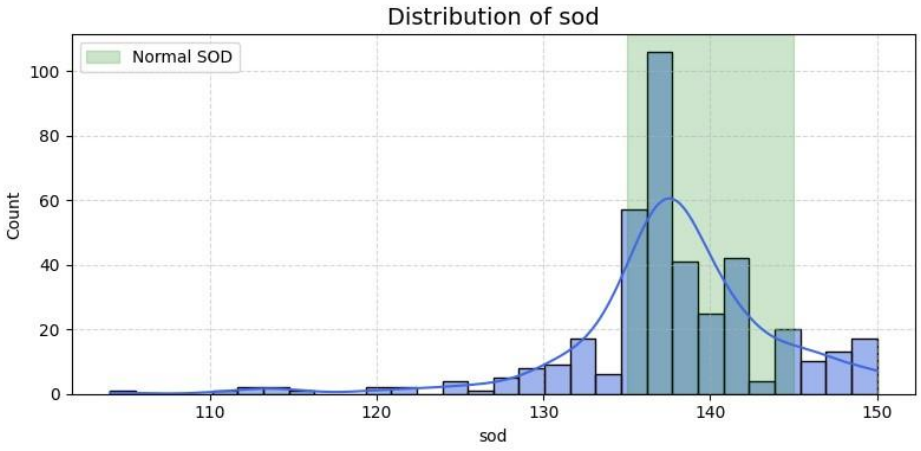
Distribution Plot with Overlay of Normal Medical Ranges:

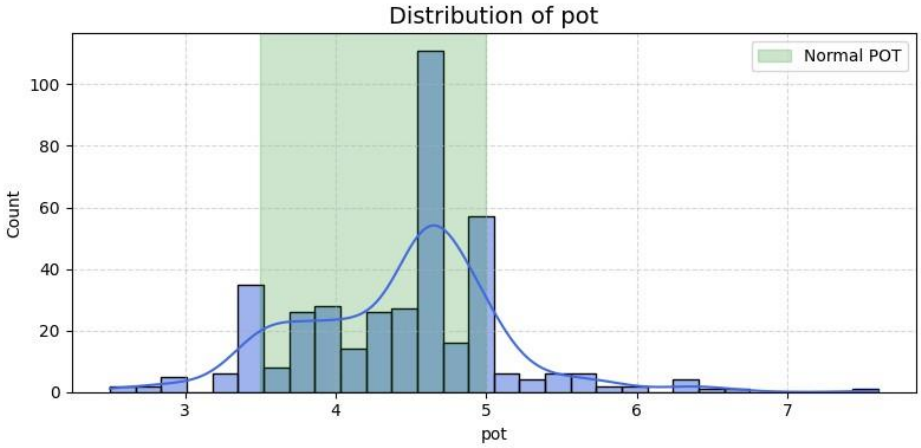

Distribution of sg
Normal SG
Count
sg
0
20
40
60
80
100
1.005
1.010
1.015
1.020
1.025
1.030

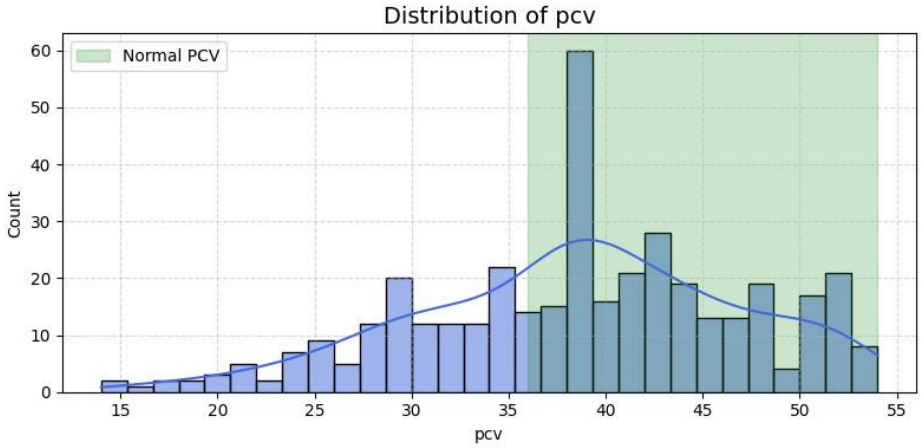

Distribution of bgr
Normal BGR
Count
bgr
0
10
20
30
40
50
60
70
0
100
200
300
400
500

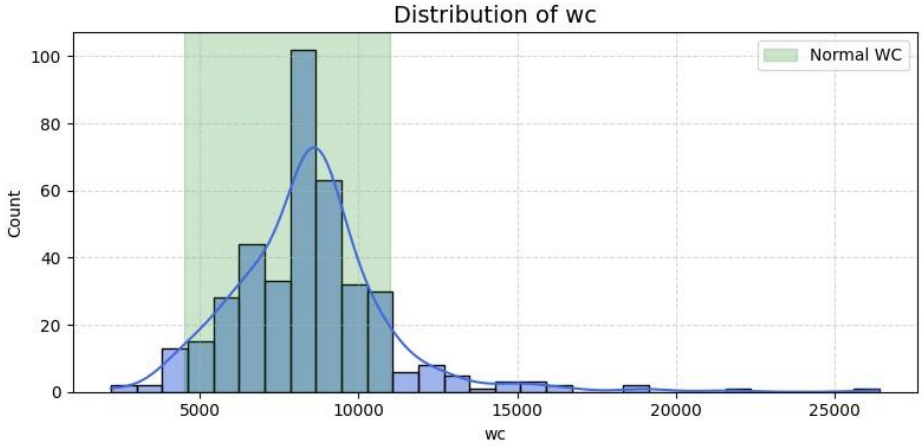

Distribution of bu
Normal BU
Count
bu
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300

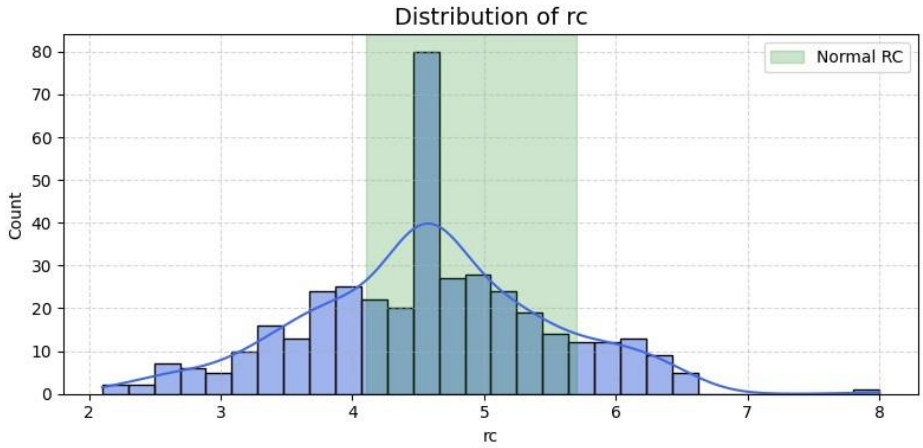

Distribution of sc
Normal SC
Count
sc
0
50
100
150
200
0
10
20
30
40
50


Distribution of sod
Normal SOD
Count
sod
0
20
40
60
80
100
110
120
130
140
150

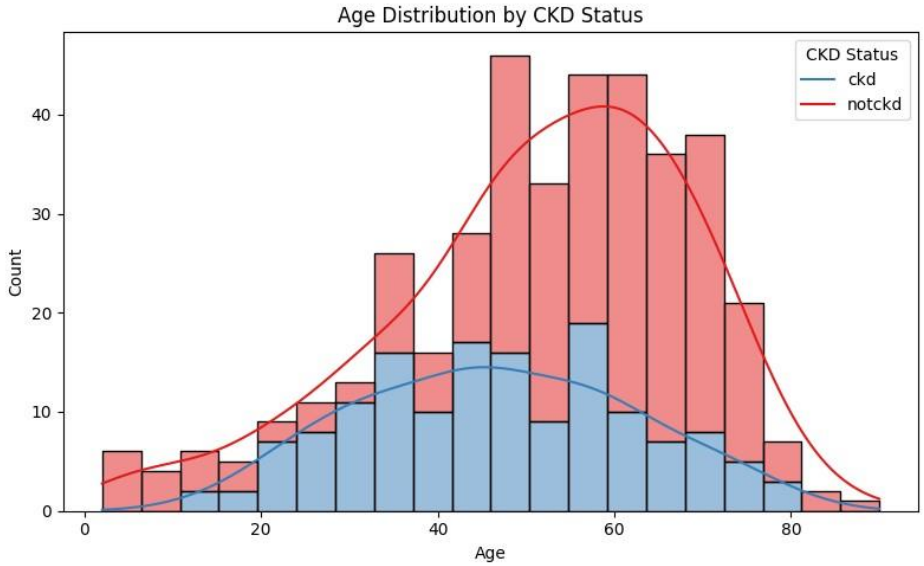

Distribution of pot
Normal POT
Count
pot
0
20
40
60
80
100
3
4
5
6
7

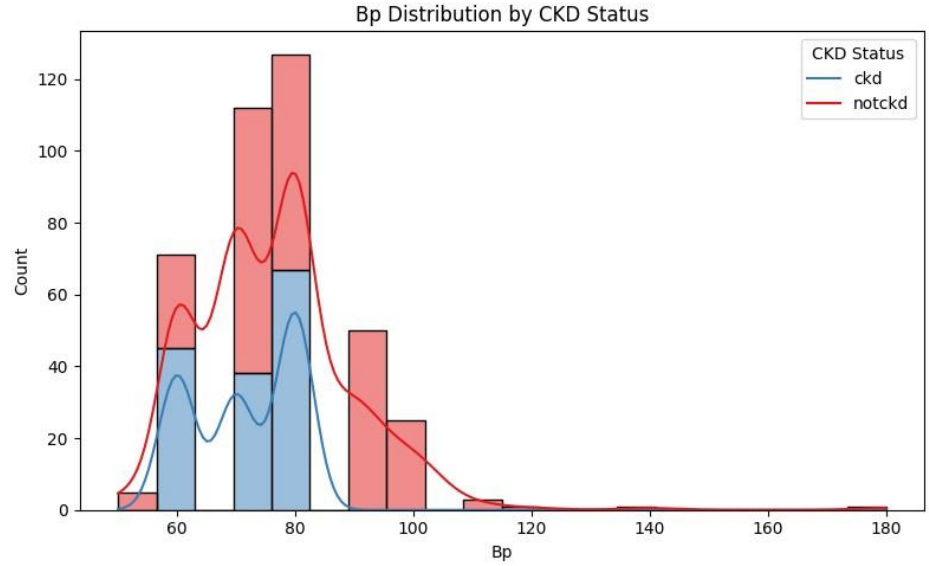

Distribution of pcv
Normal PCV
Count
pcv

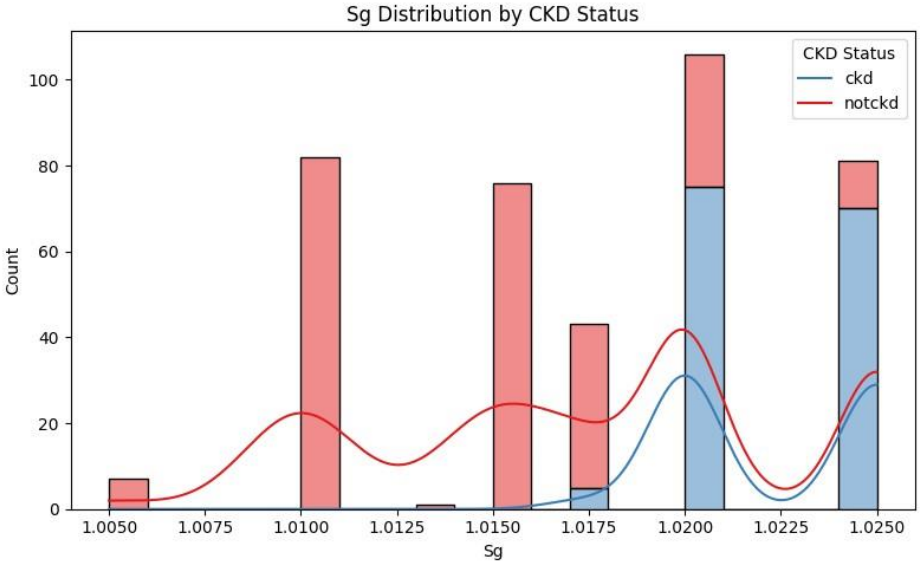

Distribution of wc
Normal WC
Count
wc

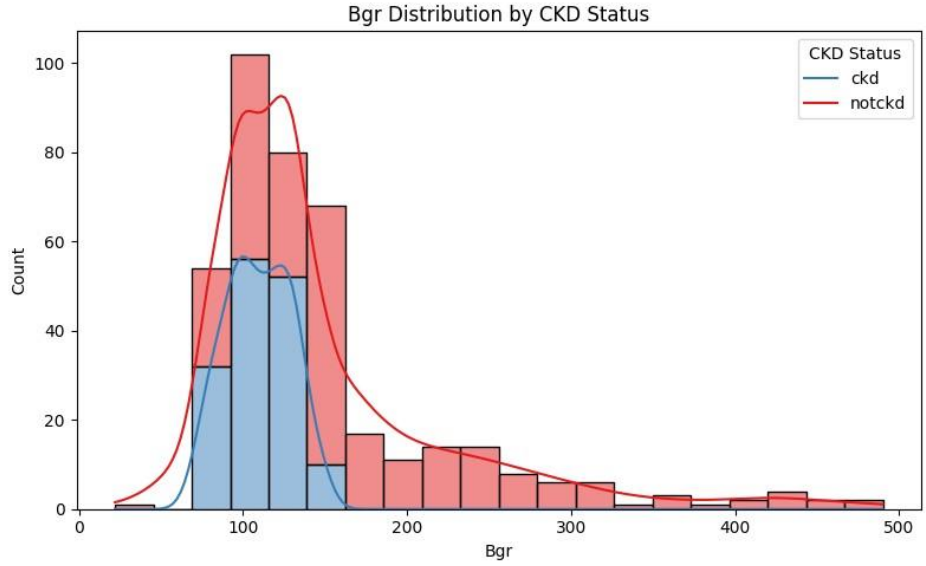

Distribution of rc
Normal RC
Count
rc

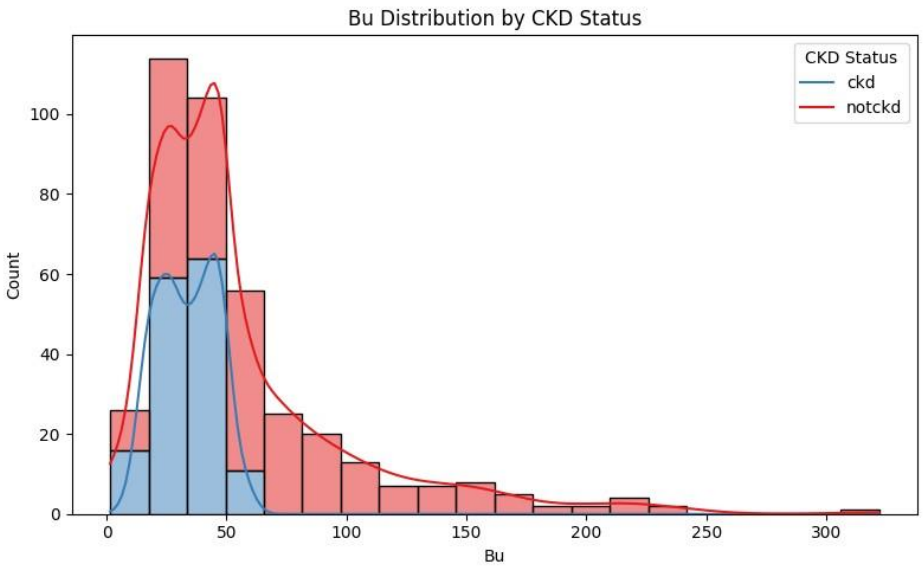
Numerical Columns with Hue for CKD Status Proportion:

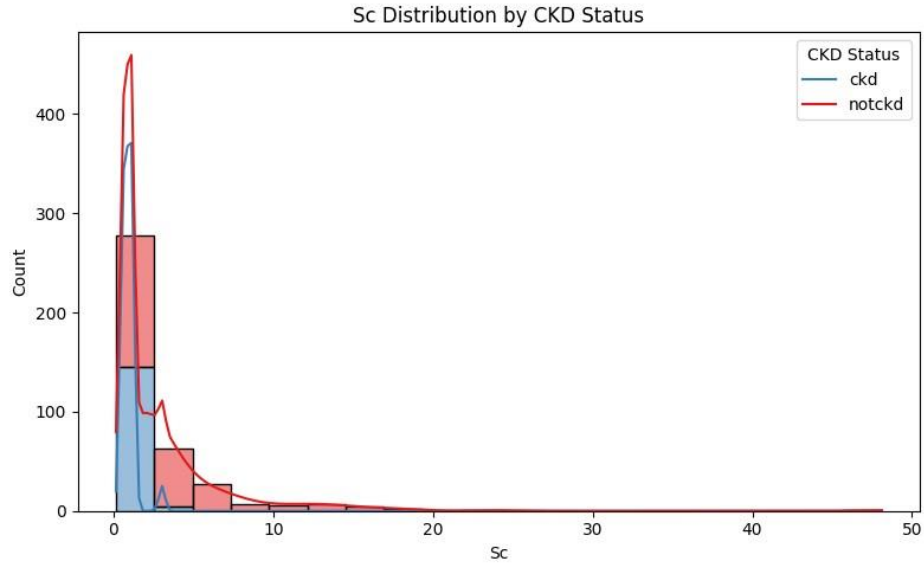


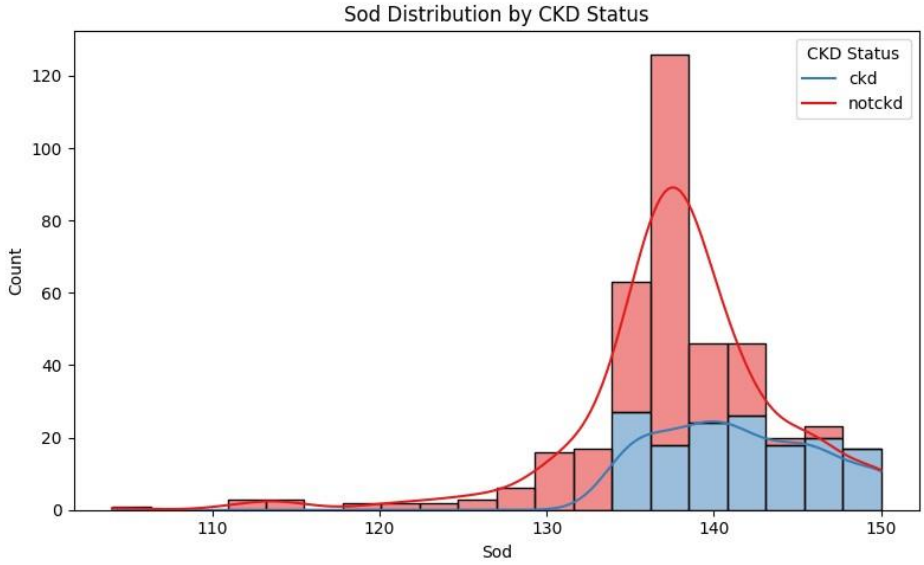


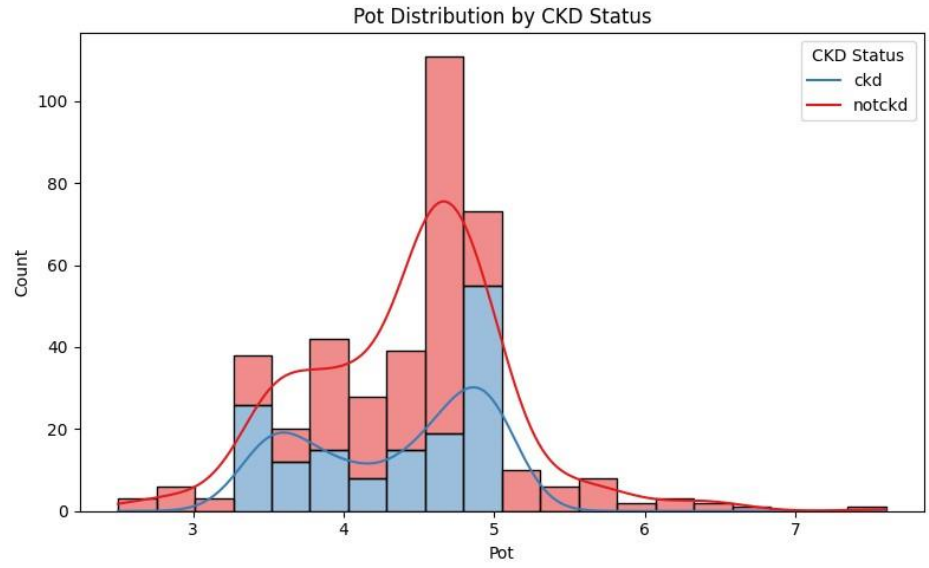

Sg Distribution by CKD Status
CKD Status
ckd
notckd
Count
Sg
0
20
40
60
80
100
1.0050
1.0075
1.0100
1.0125
1.0150
1.0175
1.0200
1.0225
1.0250

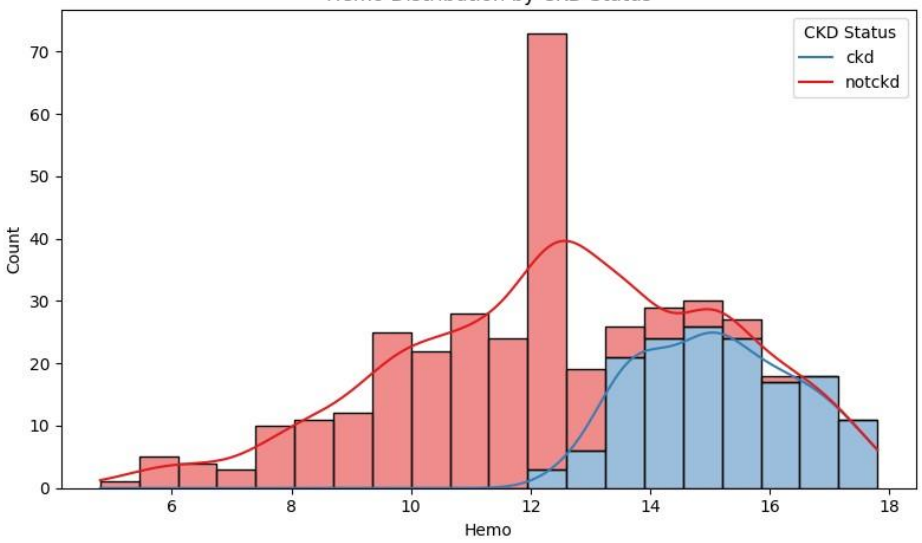

Bgr Distribution by CKD Status
CKD Status
ckd
notckd
Count
Bgr
0
20
40
60
80
100
0
100
200
300
400
500

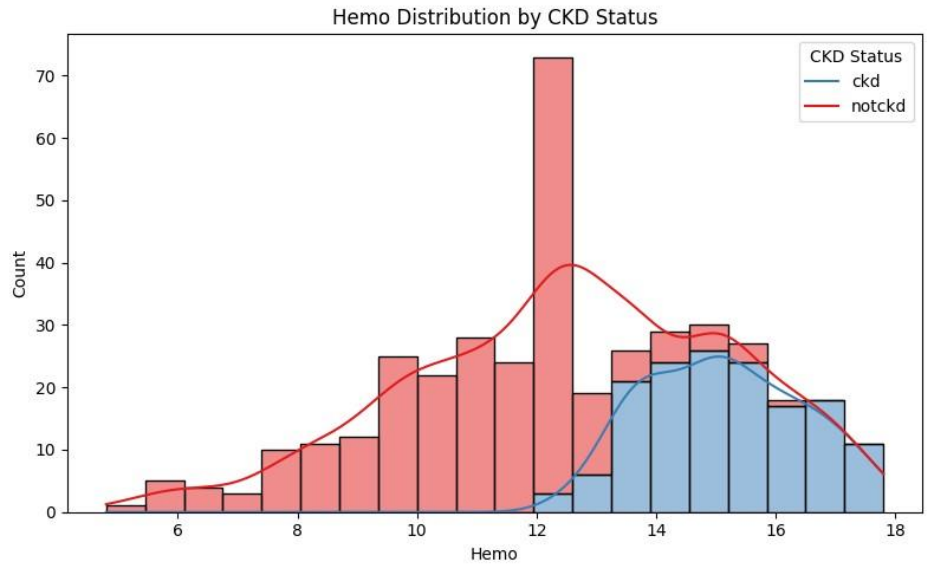

Bu Distribution by CKD Status
CKD Status
ckd
notckd
Count
Bu
0
20
40
60
80
100
0
50
100
150
200
250
300

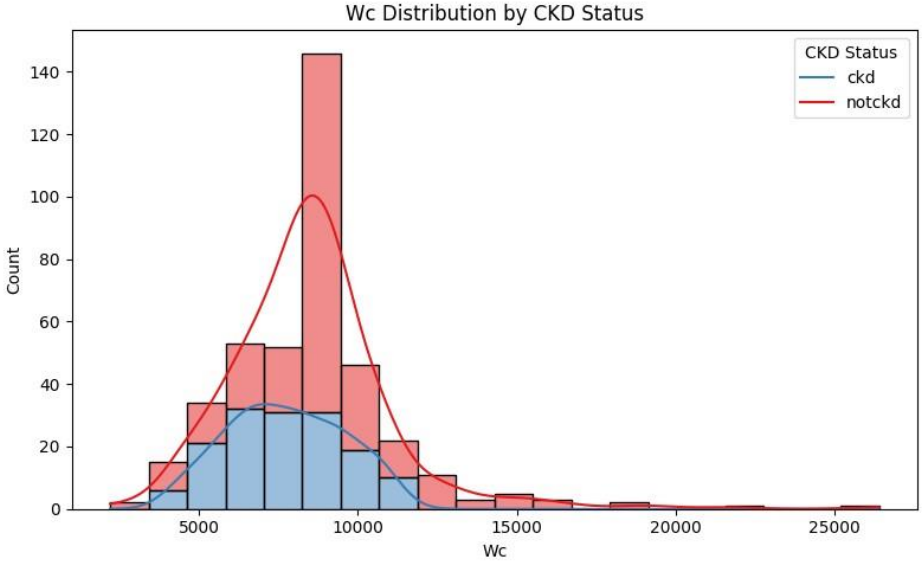

Sc Distribution by CKD Status
CKD Status
ckd
notckd
Count
Sc
0
100
200
300
400
0
10
20
30
40
50

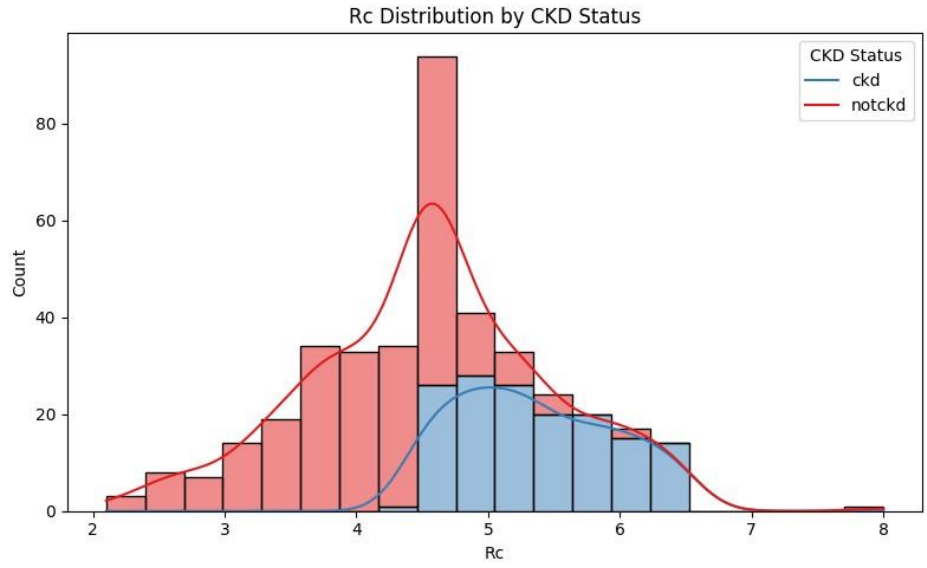

Sod Distribution by CKD Status
CKD Status
ckd
notckd
Count
Sod
0
20
40
60
80
100
120
110
120
130
140
150


Pot Distribution by CKD Status
CKD Status
ckd
notckd
Count
Pot
0
20
40
60
80
100
3
4
5
6
7

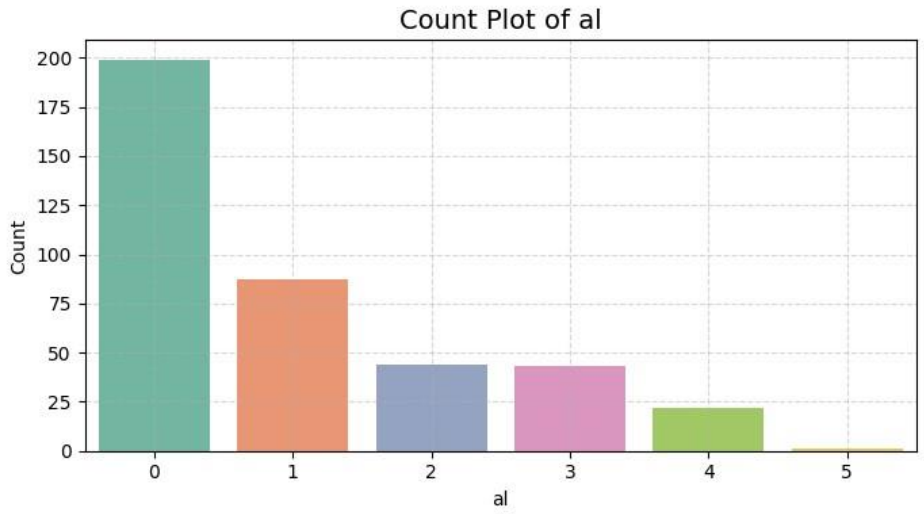

Hemo Distribution by CKD Status
CKD Status
ckd
notckd
Count
Hemo
0
10
20
30
40
50
60
70
6
8
10
12
14
16
18

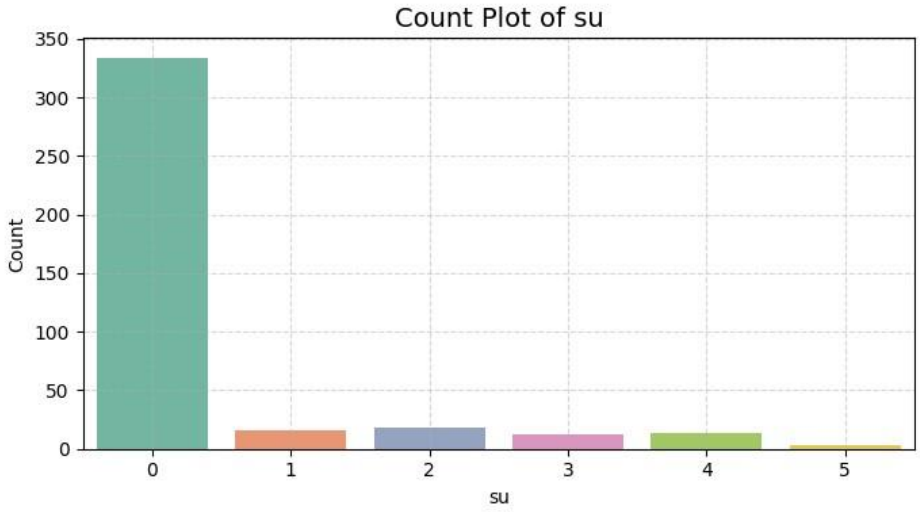

Hemo Distribution by CKD Status
CKD Status
ckd
notckd
Count
Hemo
0
10
20
30
40
50
60
70
6
8
10
12
14
16
18

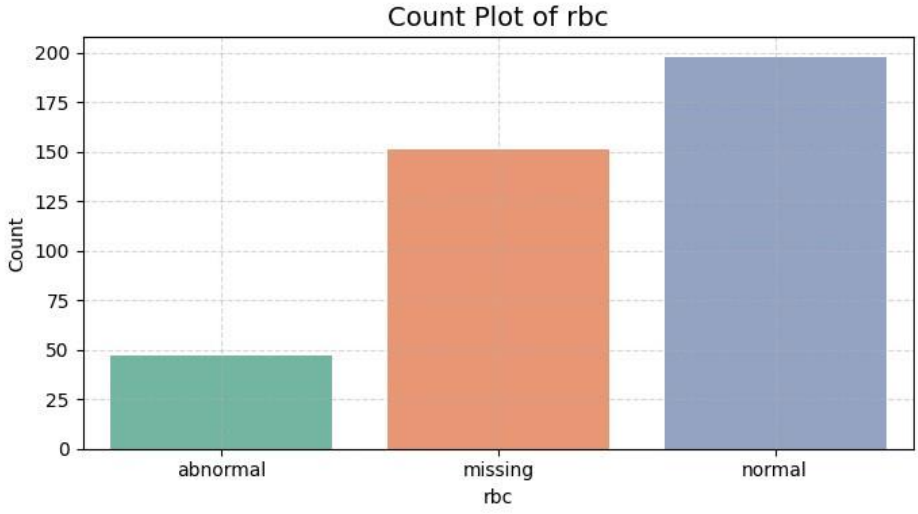

Wc Distribution by CKD Status
CKD Status
ckd
notckd
Count
Wc
0
20
40
60
80
100
120
140
5000
10000
15000
20000
25000

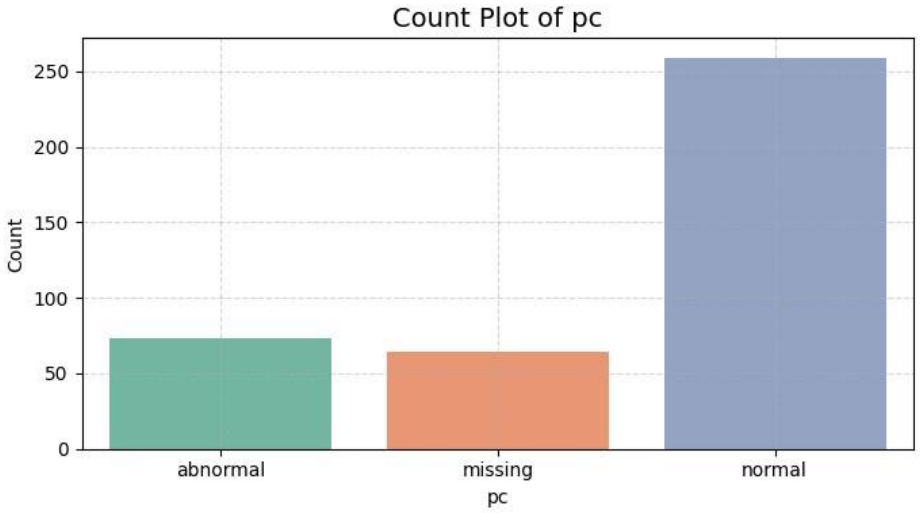

Rc Distribution by CKD Status
CKD Status
ckd
notckd
Count
Rc
0
20
40
60
80
2
3
4
5
6
7
8

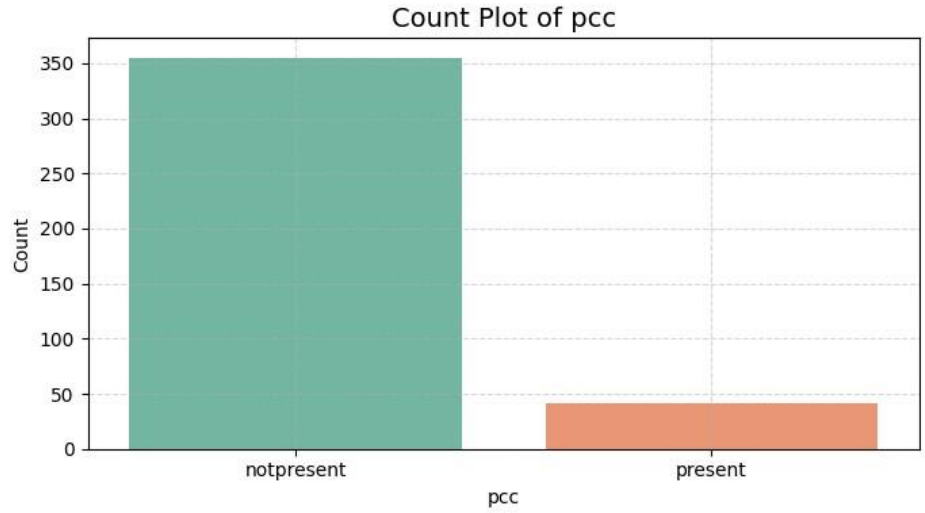
Count Plots for Categorical Columns:

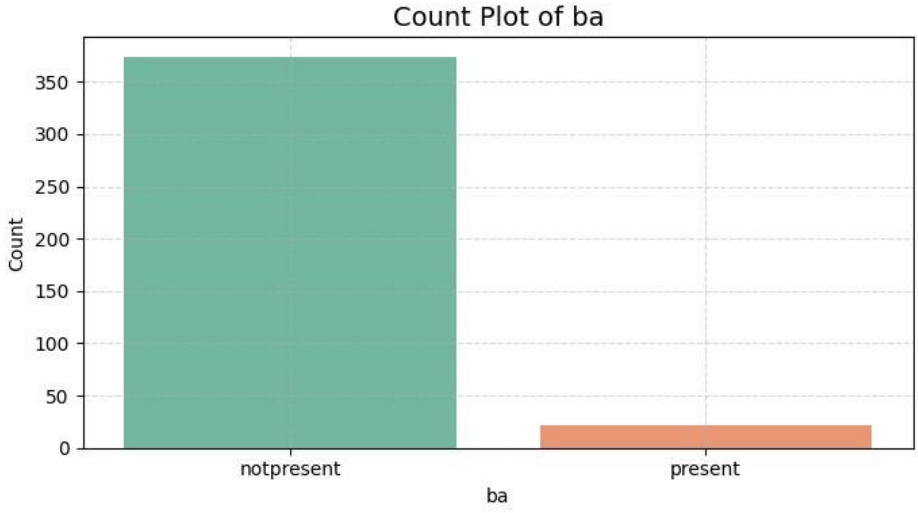


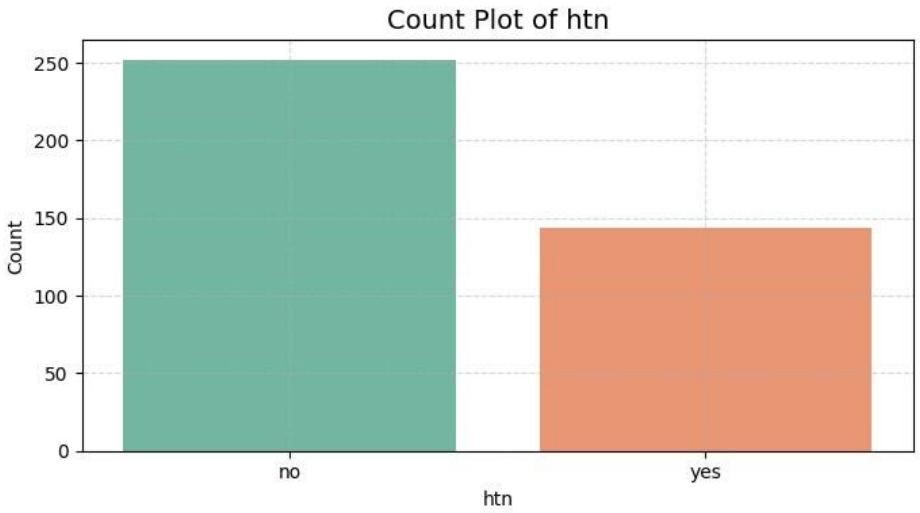


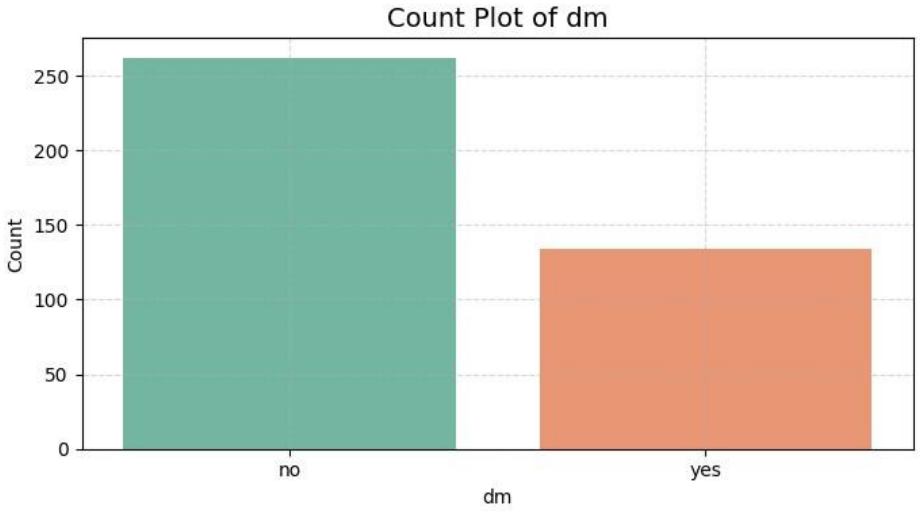


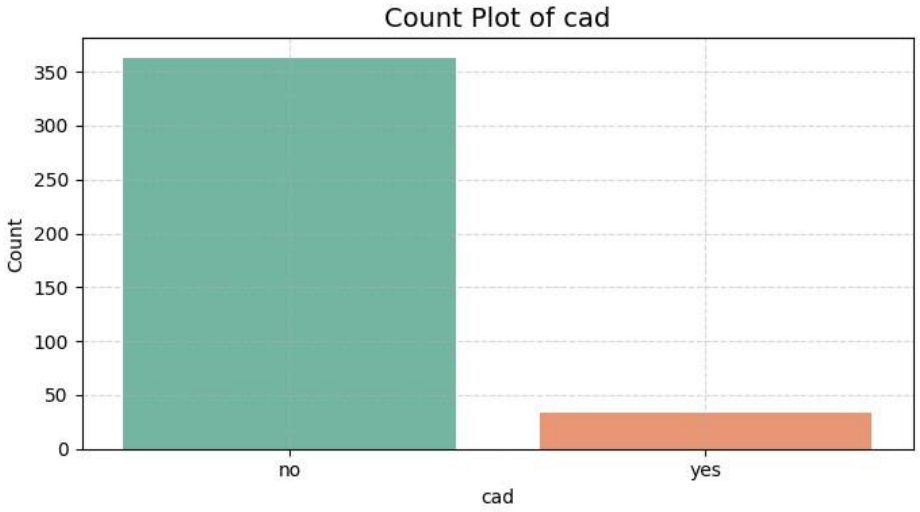

Count Plot of pc
Count
250
200
150
100
50
0
abnormal
missing
normal
pc

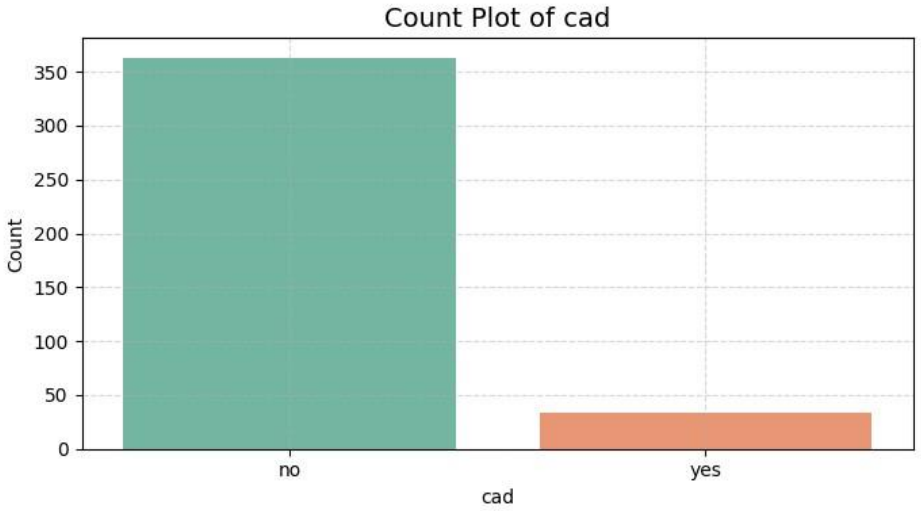

Count Plot of pcc
Count
350
300
250
200
150
100
50
0
notpresent
present
pcc

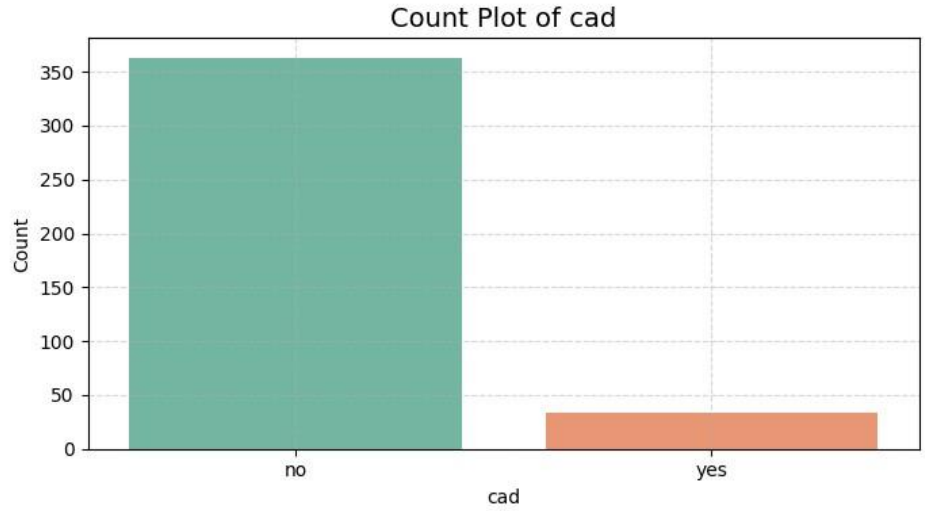

Count Plot of ba
Count
350
300
250
200
150
100
50
0
notpresent
present
ba

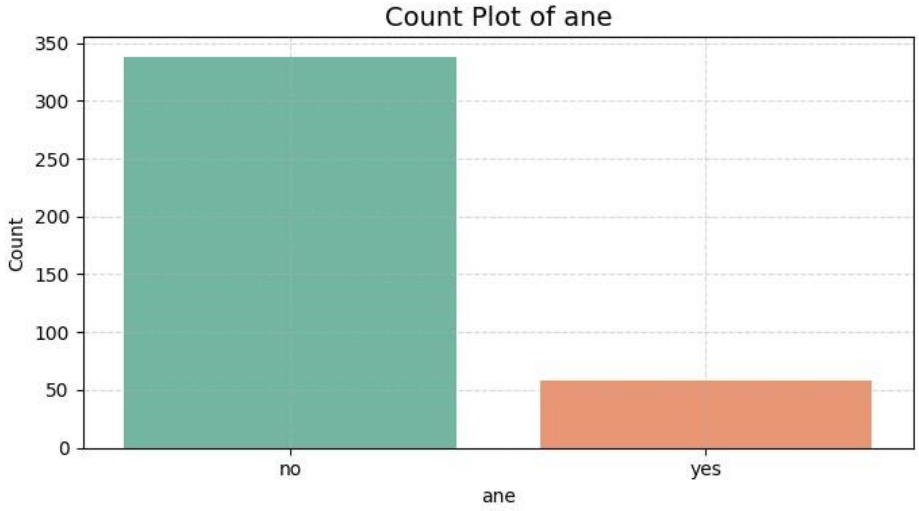

Count Plot of htn
Count
250
200
150
100
50
0
no
yes
htn

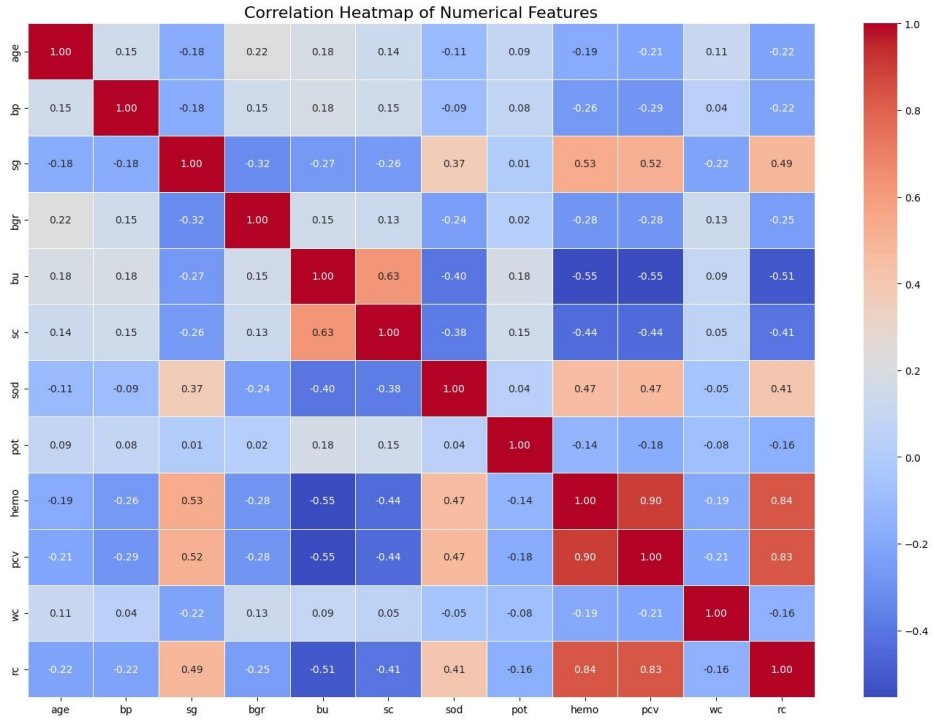

Count Plot of dm
Count
250
200
150
100
50
0
no
yes
dm


Count Plot of cad
Count
350
300
250
200
150
100
50
0
no
yes
cad

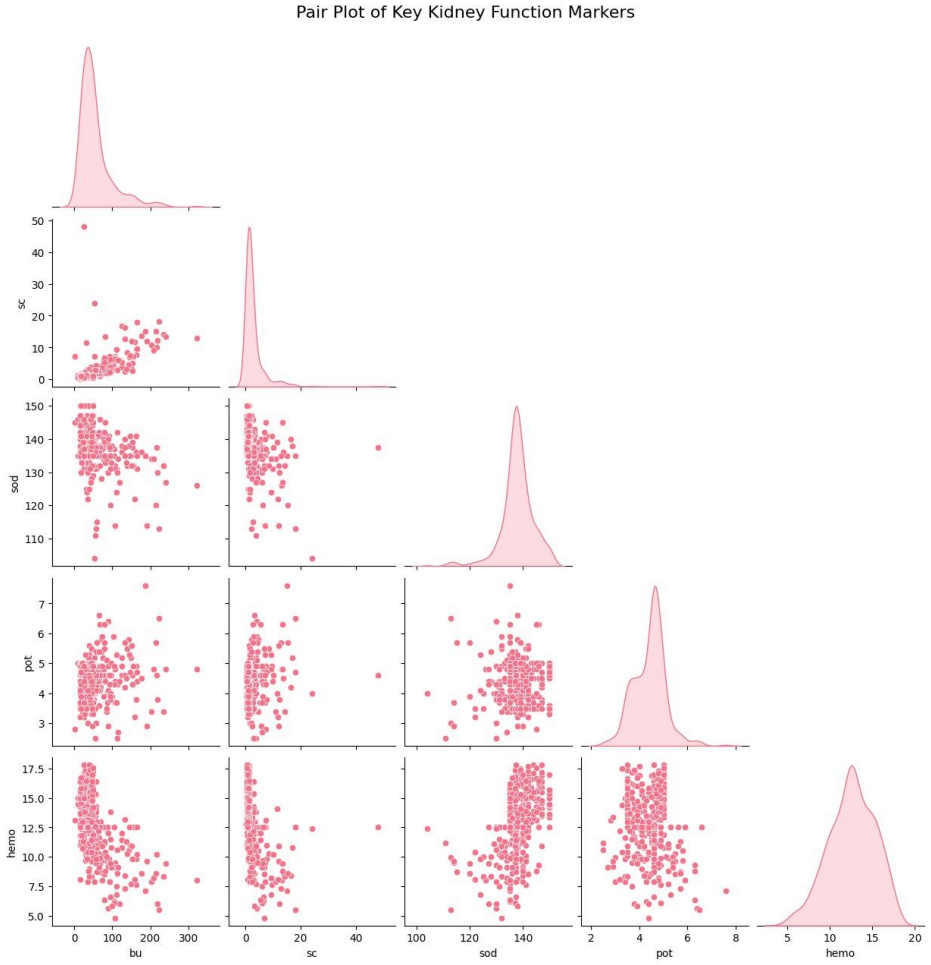

Count Plot of cad
Count
350
300
250
200
150
100
50
0
no
yes
cad


Count Plot of cad
Count
350
300
250
200
150
100
50
0
no
yes
cad

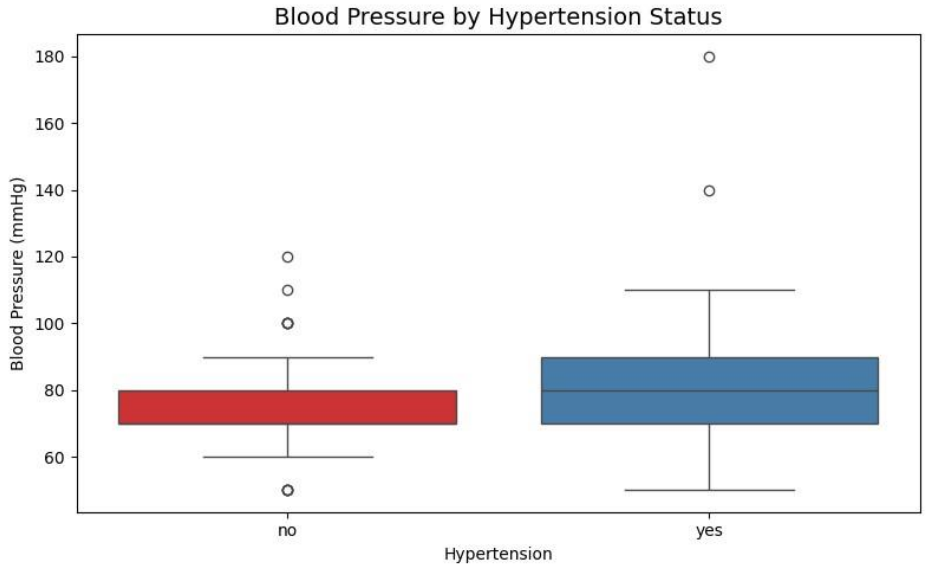

Count Plot of ane
Count
350
300
250
200
150
100
50
0
no
yes
ane

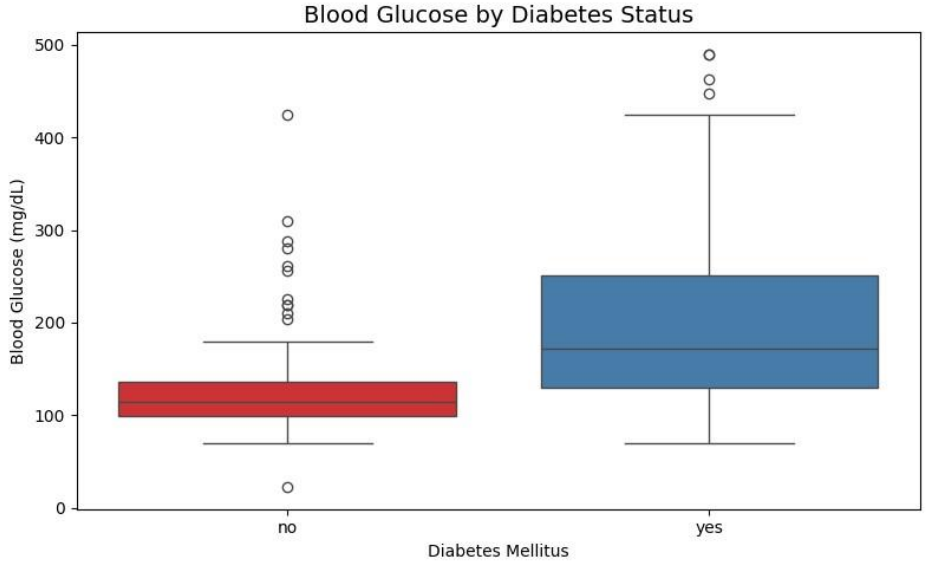
Bivariate Analysis	Correlation Heatmap for Numerical Variables: 

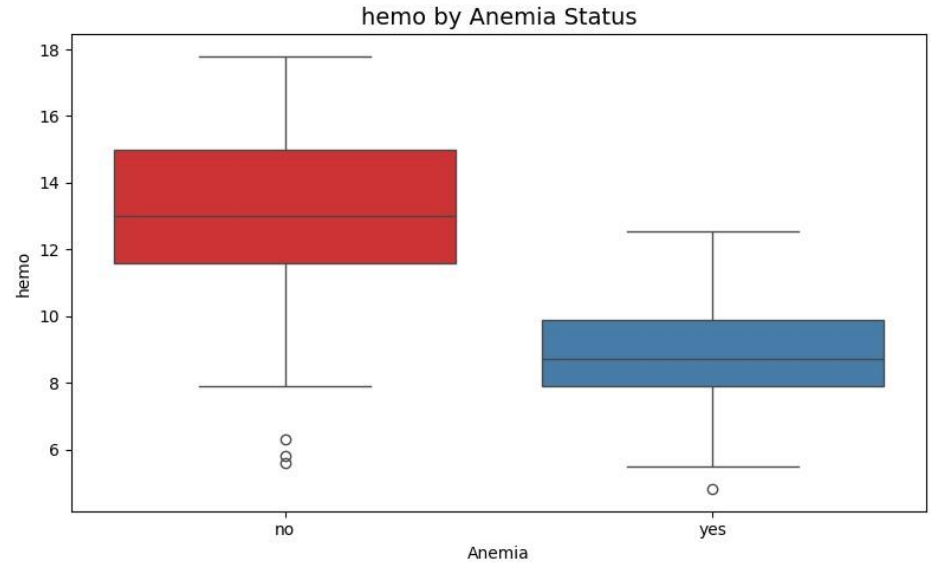
Pair Plots for Key Kidney Function Markers:

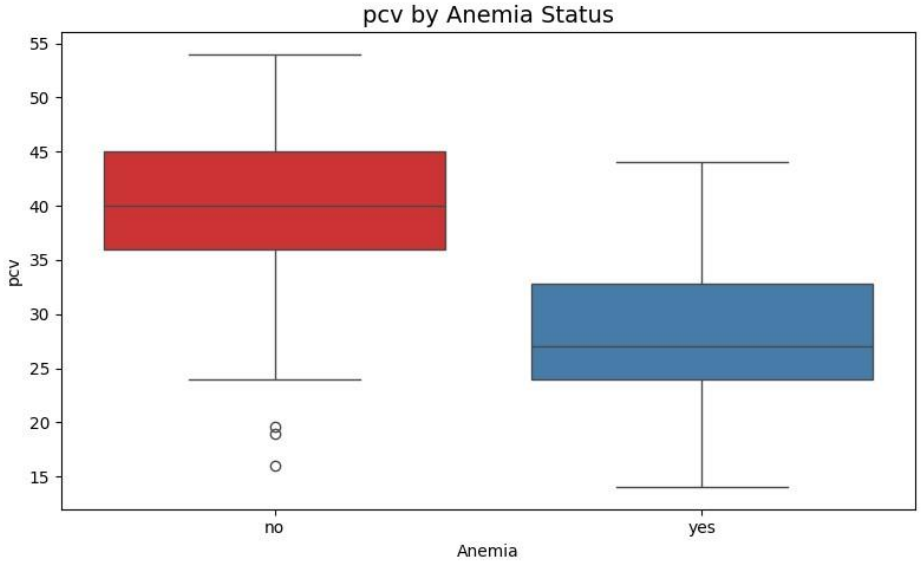


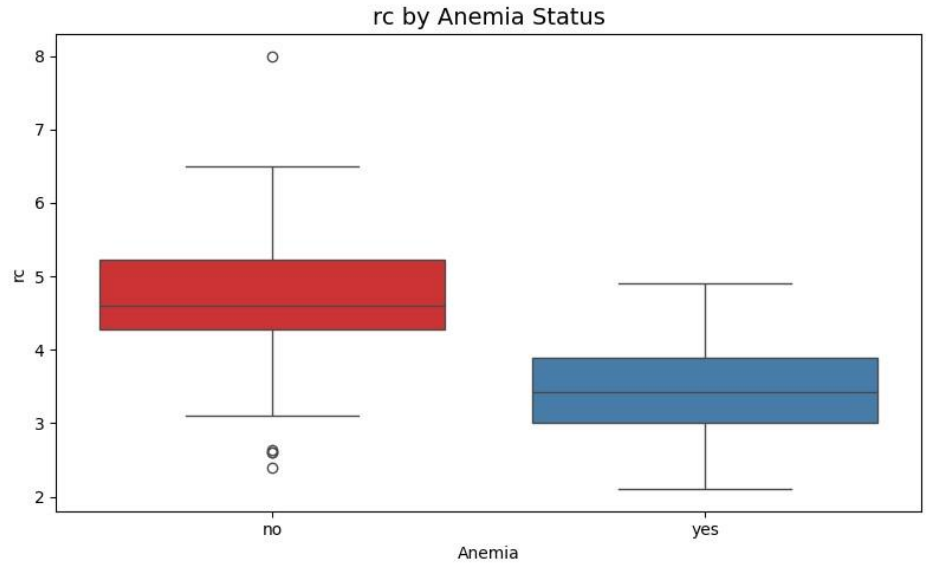
Feature Relationships with Medical Context:

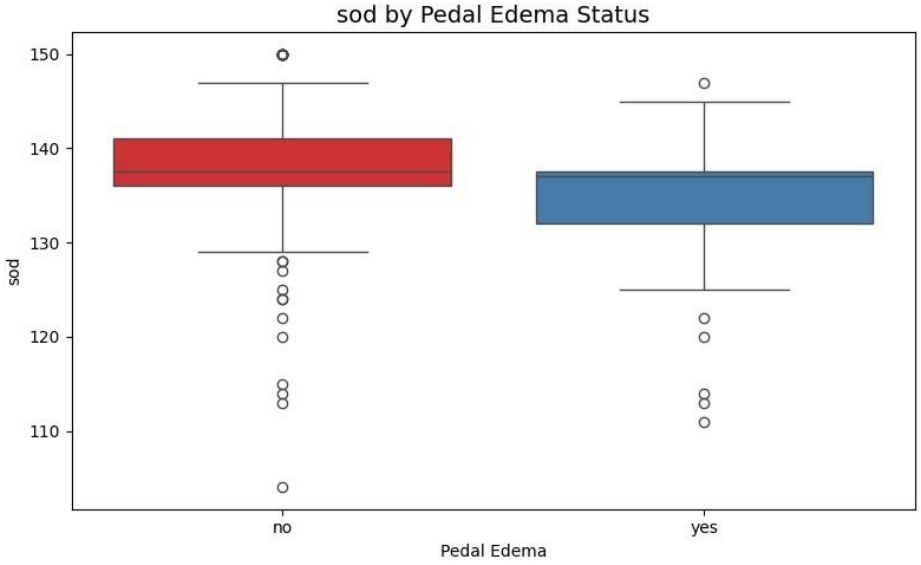


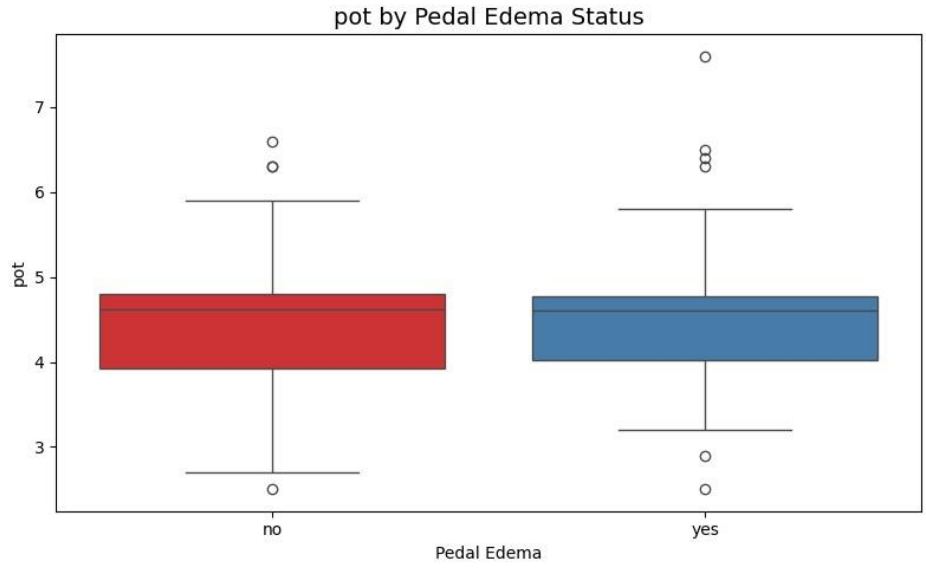

Blood Glucose by Diabetes Status
Blood Glucose (mg/dL)
500
400
300
200
100
0
no
yes
Diabetes Mellitus

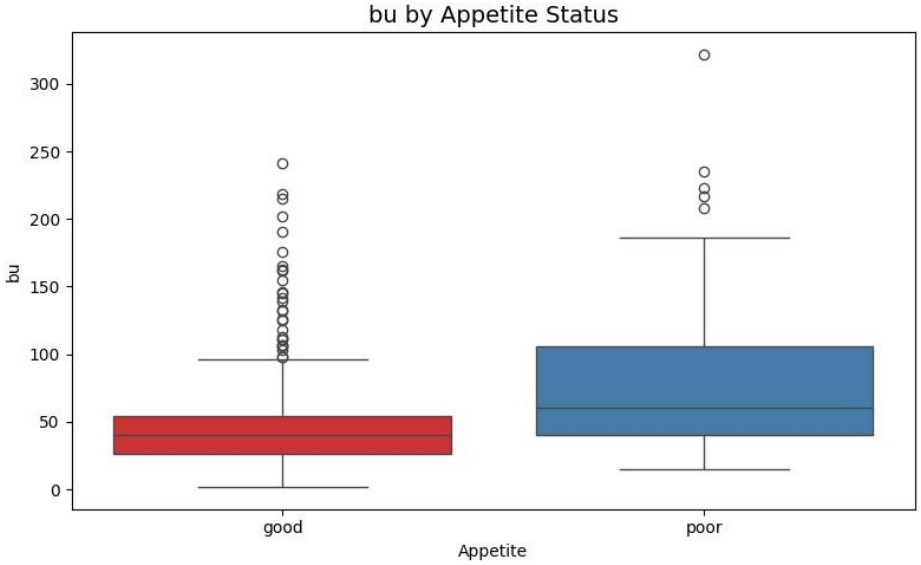

hemo by Anemia Status
hemo
18
16
14
12
10
8
6
no
yes
Anemia

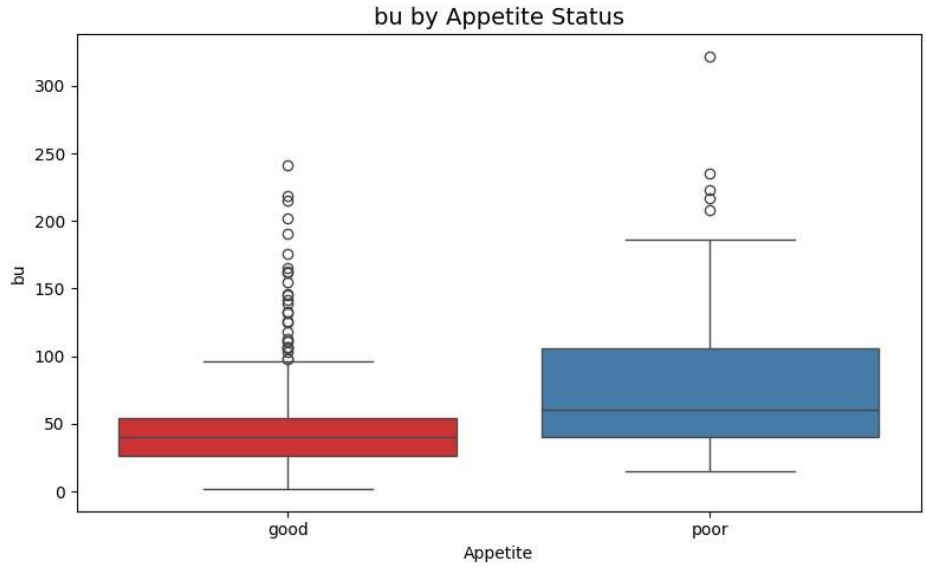

pcv by Anemia Status
pcv
55
50
45
40
35
30
25
20
15
no
yes
Anemia


rc by Anemia Status
rc
8
7
6
5
4
3
2
no
yes
Anemia

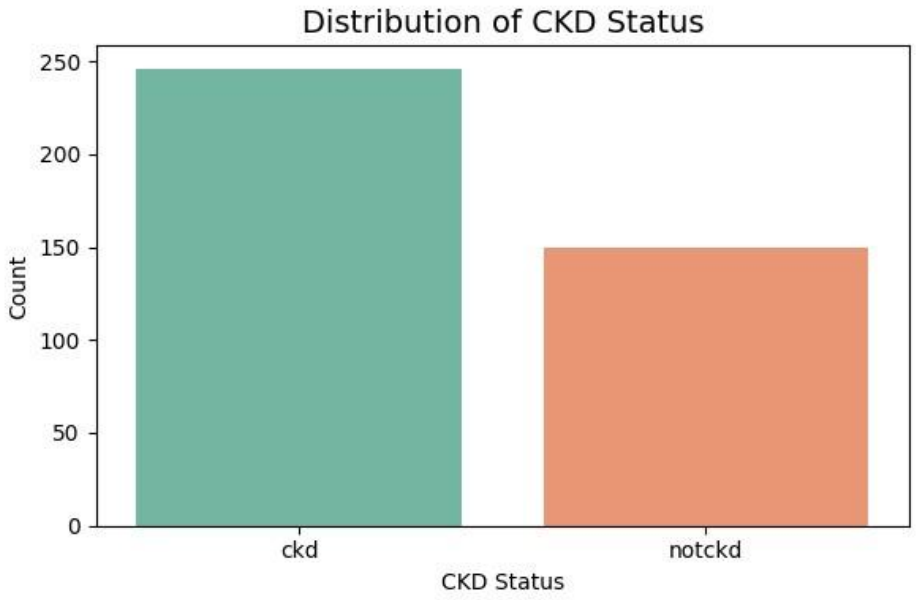

sod by Pedal Edema Status
150
140
130
120
110
sod
no
yes
Pedal Edema


pot by Pedal Edema Status
7
6
5
4
3
pot
no
yes
Pedal Edema

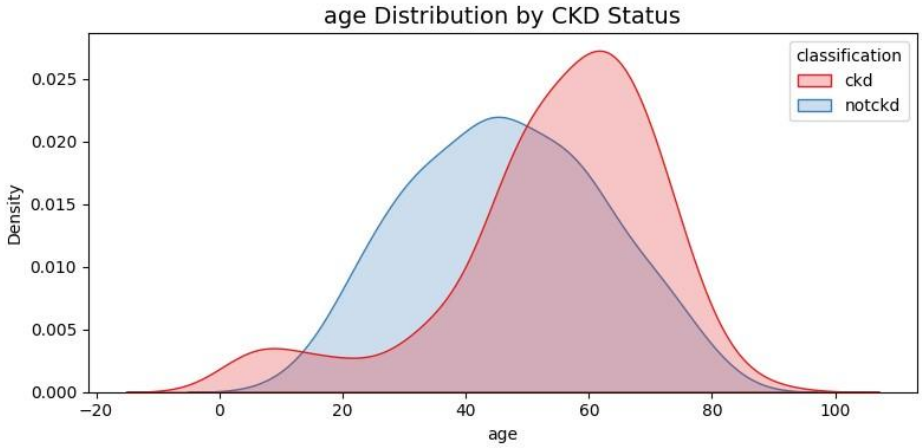

bu by Appetite Status
bu
300
250
200
150
100
50
0
good
poor
Appetite

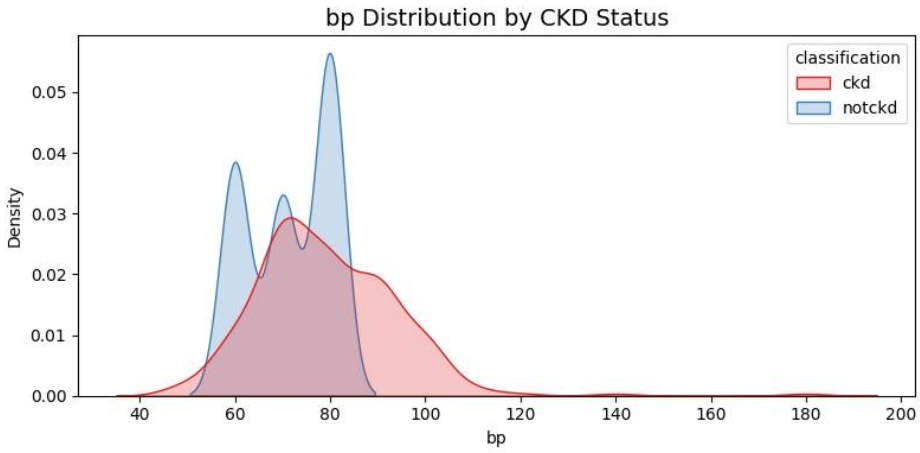

bu by Appetite Status
bu
300
250
200
150
100
50
0
good
poor
Appetite

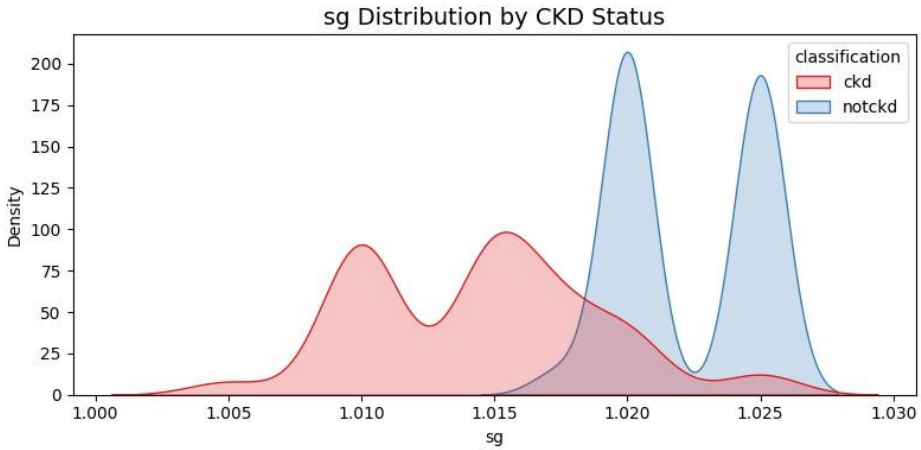
Distribution of CKD Status:

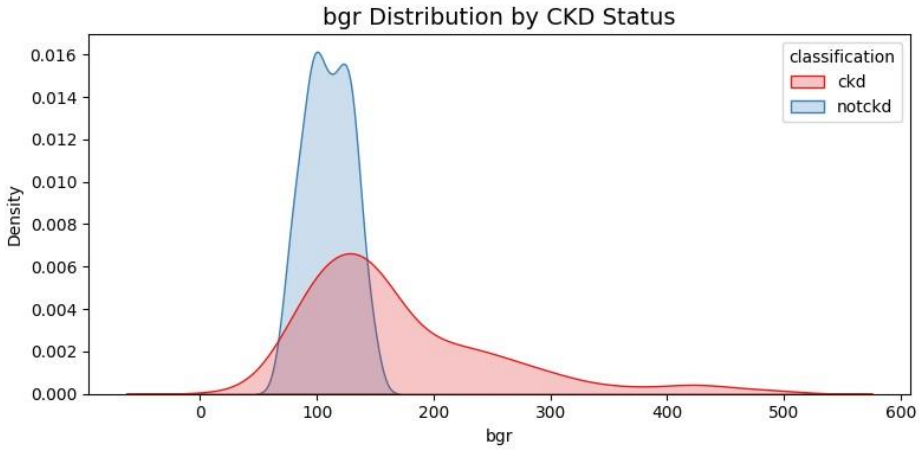


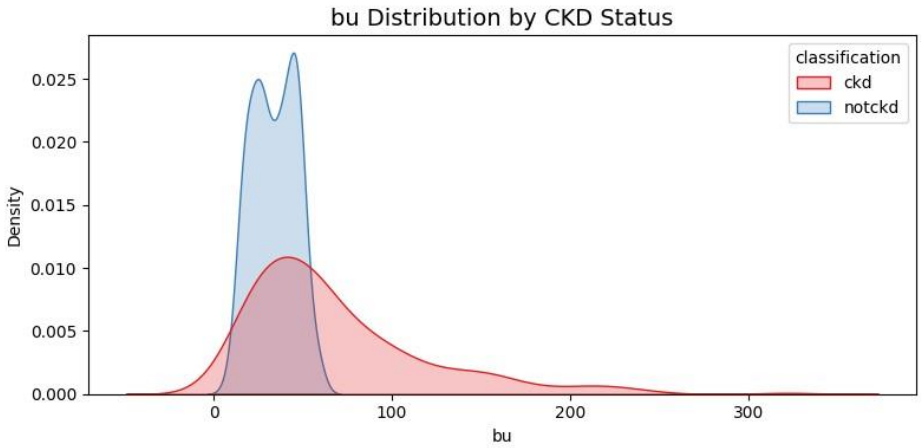
Feature Distributions by CKD Status:

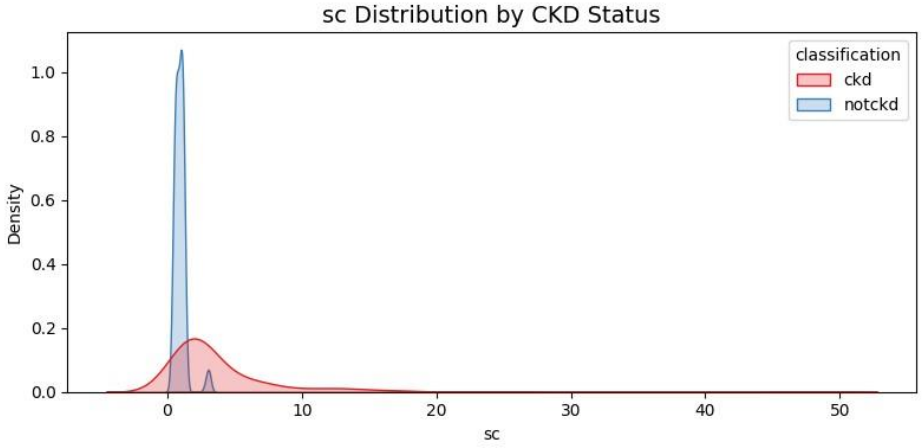


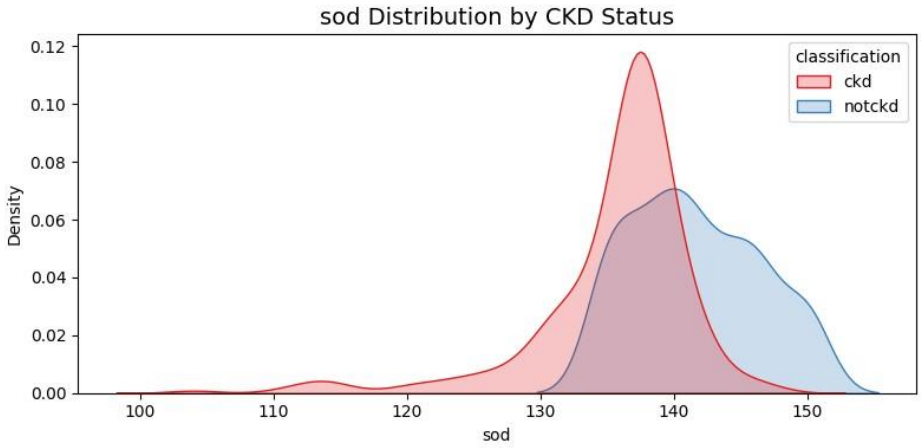

bp Distribution by CKD Status
classification
ckd
notckd
Density
0.00
0.01
0.02
0.03
0.04
0.05
40
60
80
100
120
140
160
180
200
bp

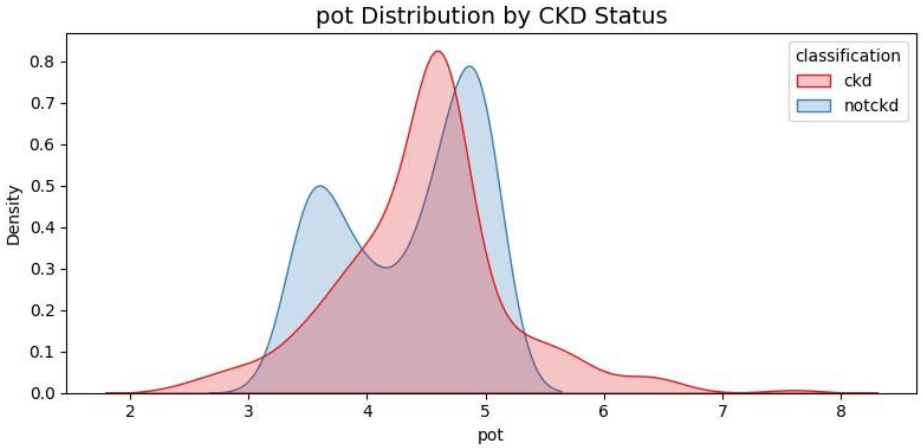

sg Distribution by CKD Status
classification
ckd
notckd
Density
0
25
50
75
100
125
150
175
200
1.000
1.005
1.010
1.015
1.020
1.025
1.030
sg

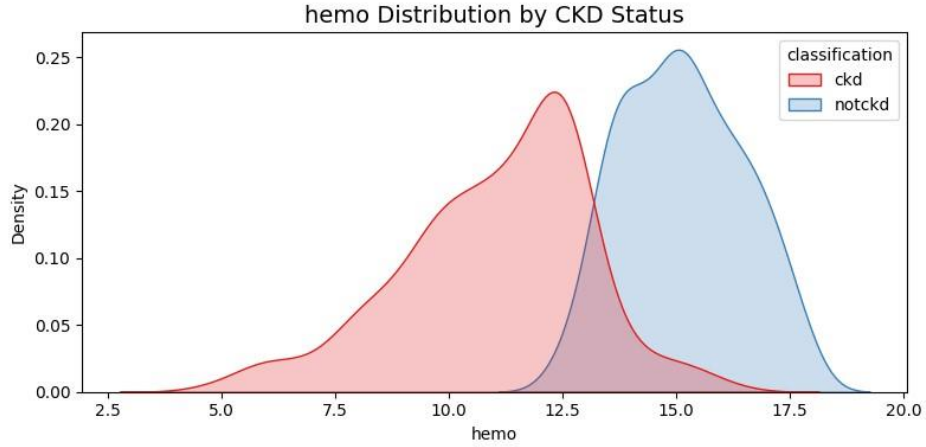

bgr Distribution by CKD Status
classification
ckd
notckd
Density
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0
100
200
300
400
500
600
bgr

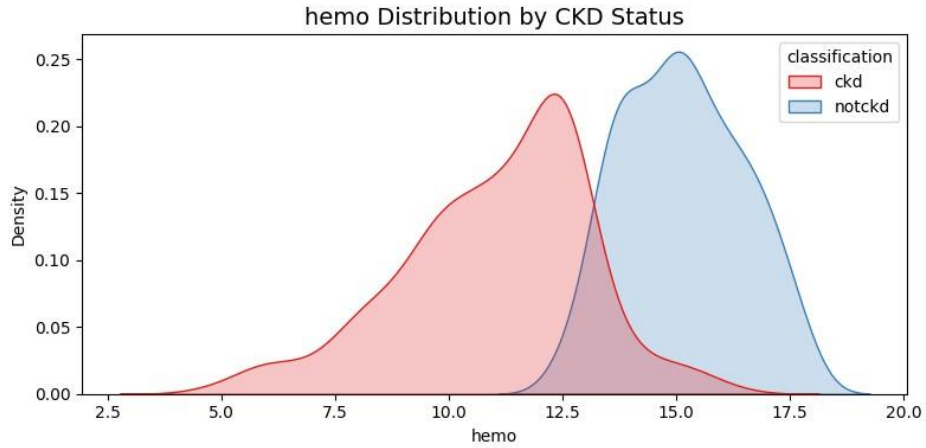

bu Distribution by CKD Status
classification
ckd
notckd
Density
0.025
0.020
0.015
0.010
0.005
0.000
0
100
200
300
bu

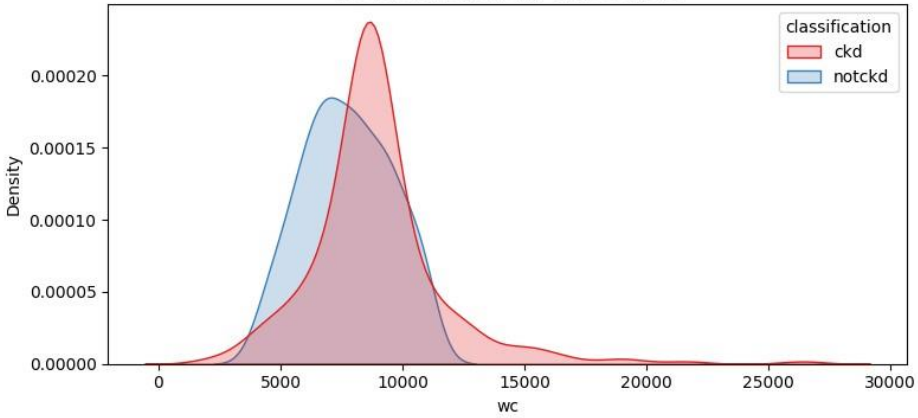

sc Distribution by CKD Status
classification
ckd
notckd
Density
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
sc

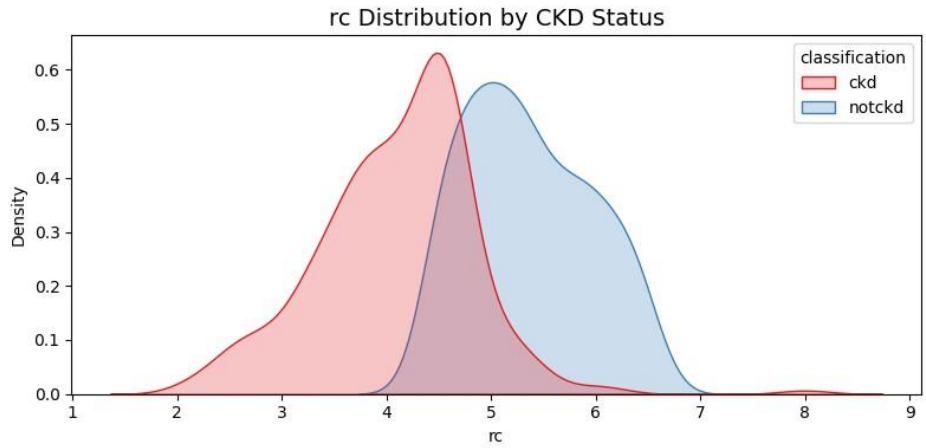

sod Distribution by CKD Status
classification
ckd
notckd
Density
0.12
0.10
0.08
0.06
0.04
0.02
0.00
100
110
120
130
140
150
sod

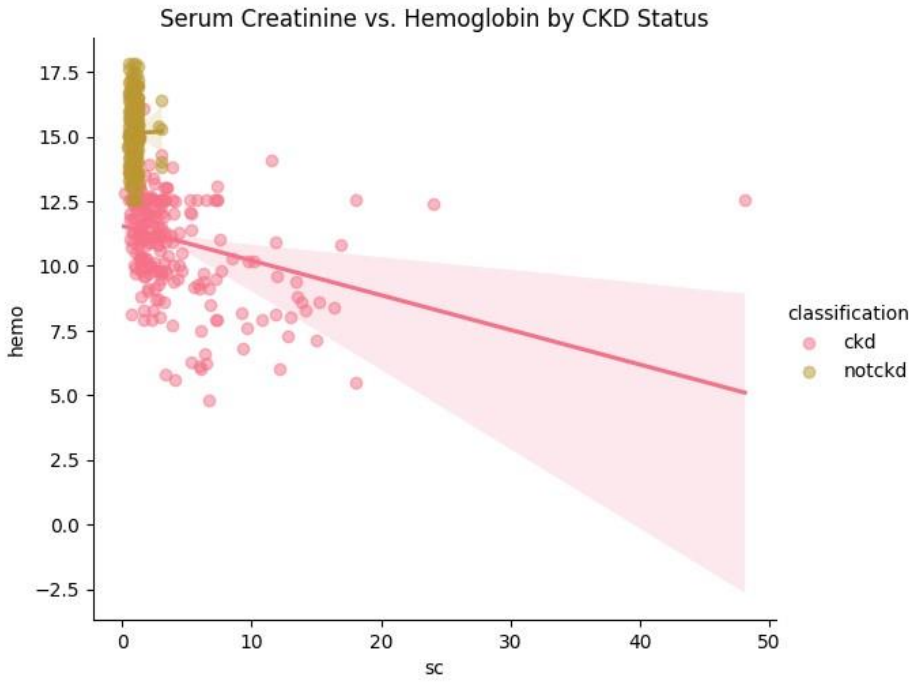
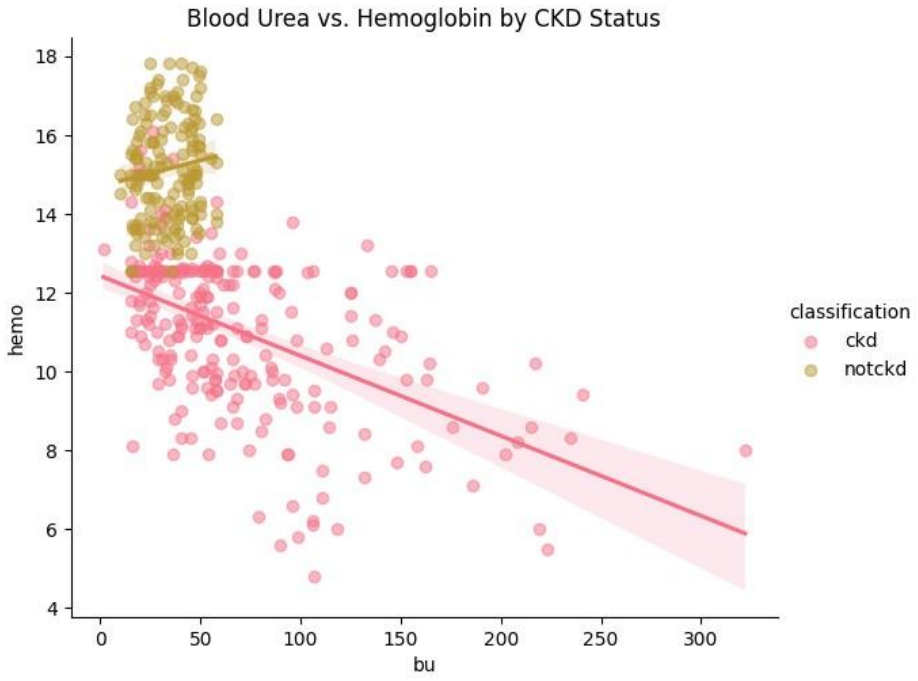

pot Distribution by CKD Status
classification
ckd
notckd
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
2
3
4
5
6
7
8
pot

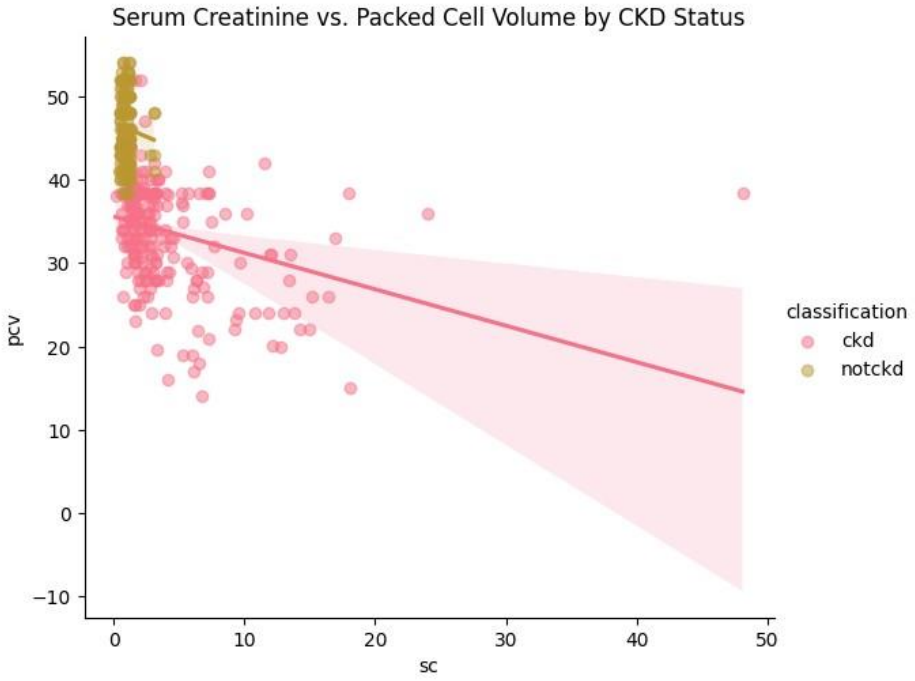

hemo Distribution by CKD Status
classification
ckd
notckd
Density
0.00
0.05
0.10
0.15
0.20
0.25
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
hemo

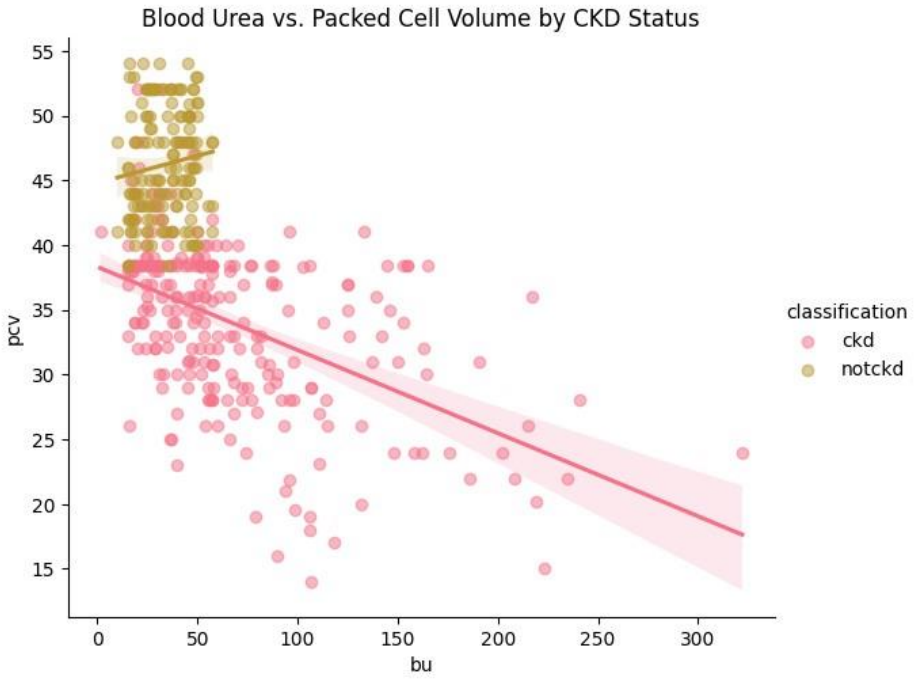

hemo Distribution by CKD Status
classification
ckd
notckd
Density
0.00
0.05
0.10
0.15
0.20
0.25
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
hemo

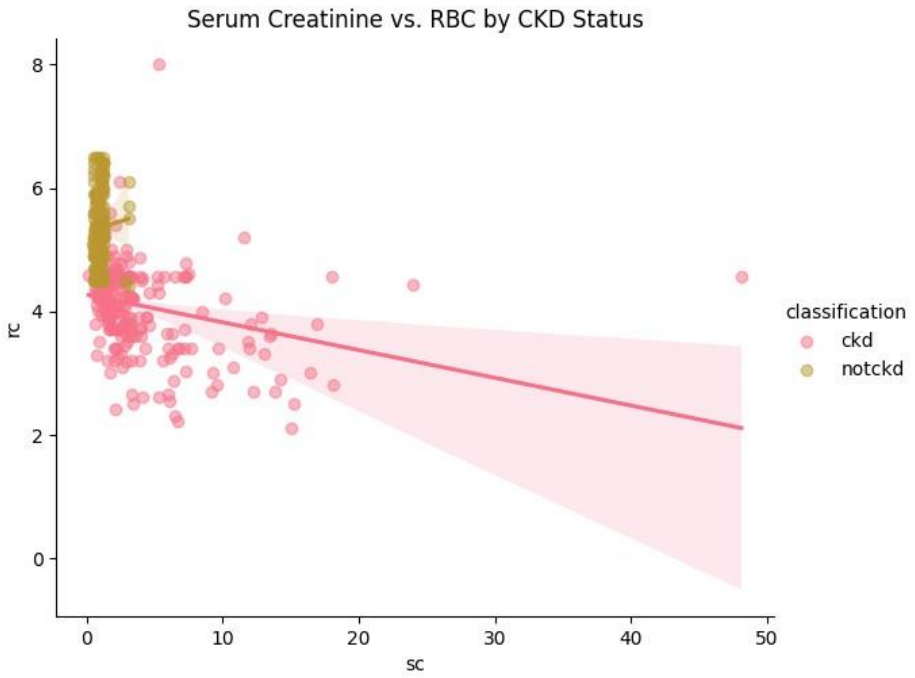

wc Distribution by CKD Status
classification
ckd
notckd
Density
0.00020
0.00015
0.00010
0.00005
0.00000
0
5000
10000
15000
20000
25000
30000
wc

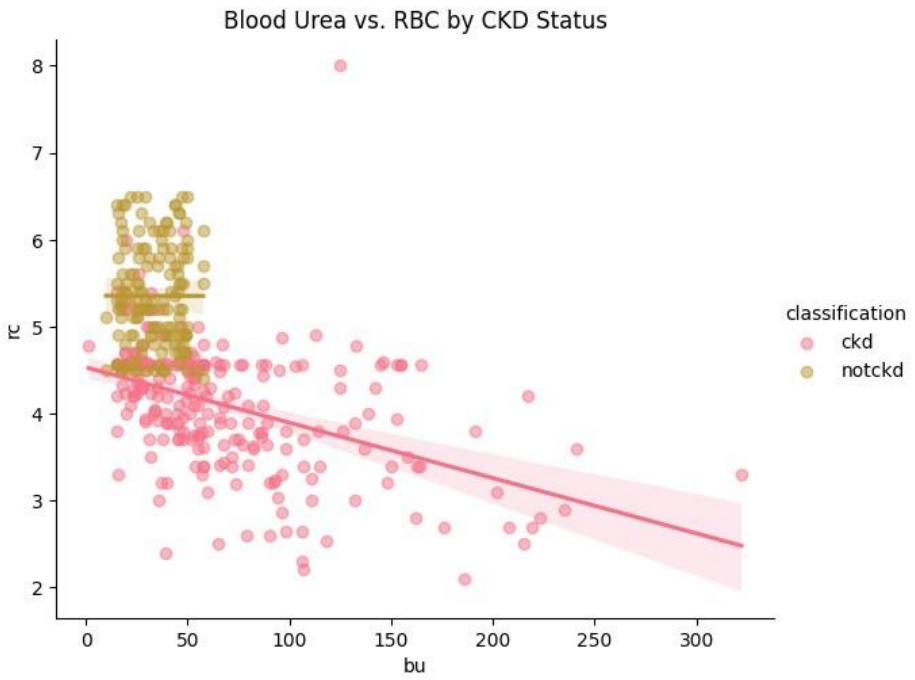

rc Distribution by CKD Status
classification
ckd
notckd
Density
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6
7
8
9
rc

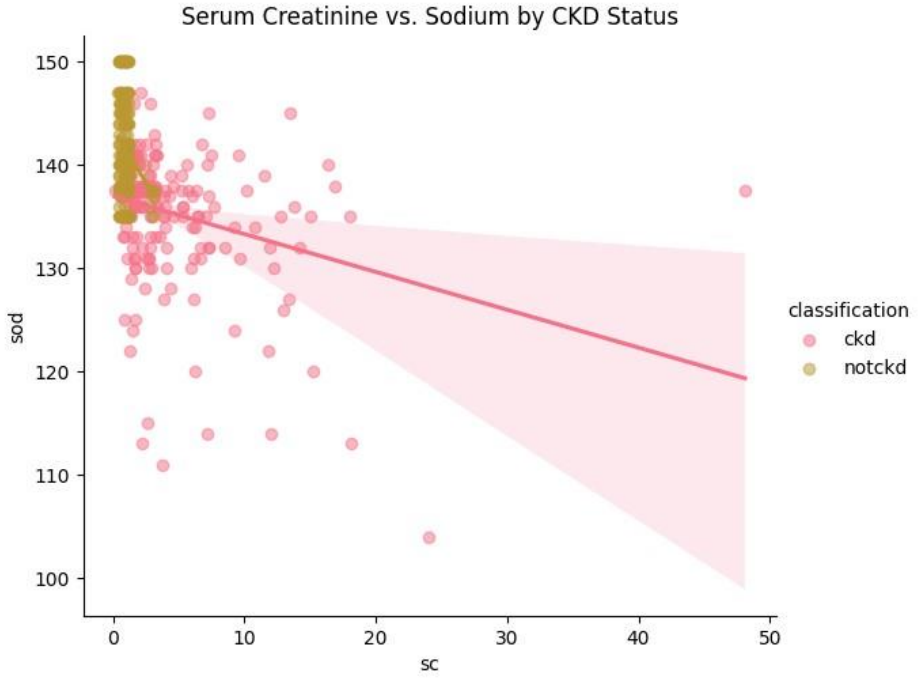
Multivariate Analysis	In Depth Analysis Based on Correlations: a. Kidney Function and Anemia • SC, BU vs. Hemo, PCV, RC: • These relationships are central to CKD progression (as kidney function declines, anemia worsens).  

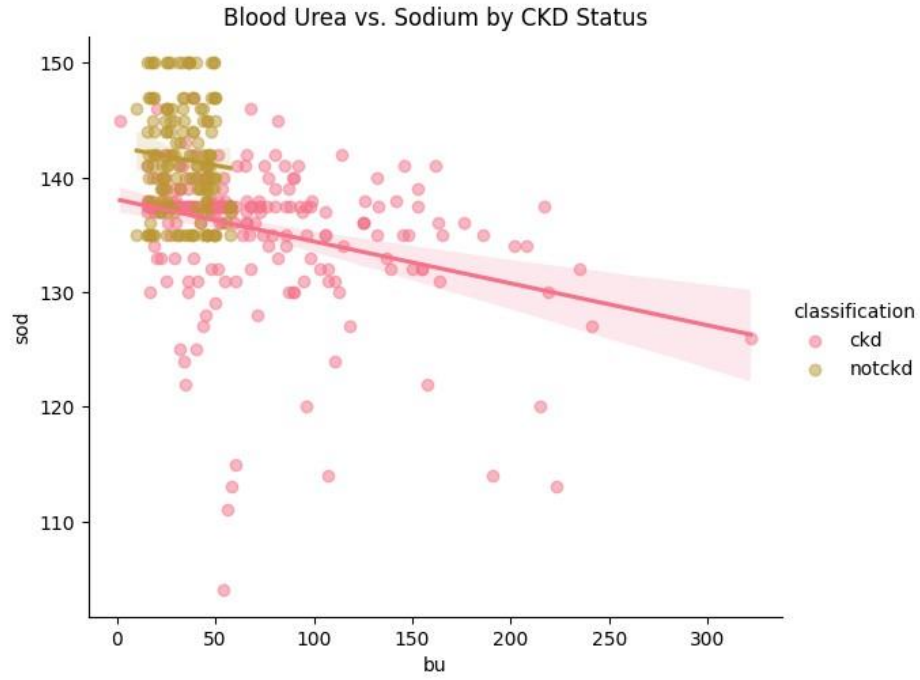

Serum Creatinine vs. Packed Cell Volume by CKD Status
pcv
sc
classification
ckd
notckd


Blood Urea vs. Packed Cell Volume by CKD Status
pcv
bu
classification
ckd
notckd


Serum Creatinine vs. RBC by CKD Status
rc
sc
classification
ckd
notckd

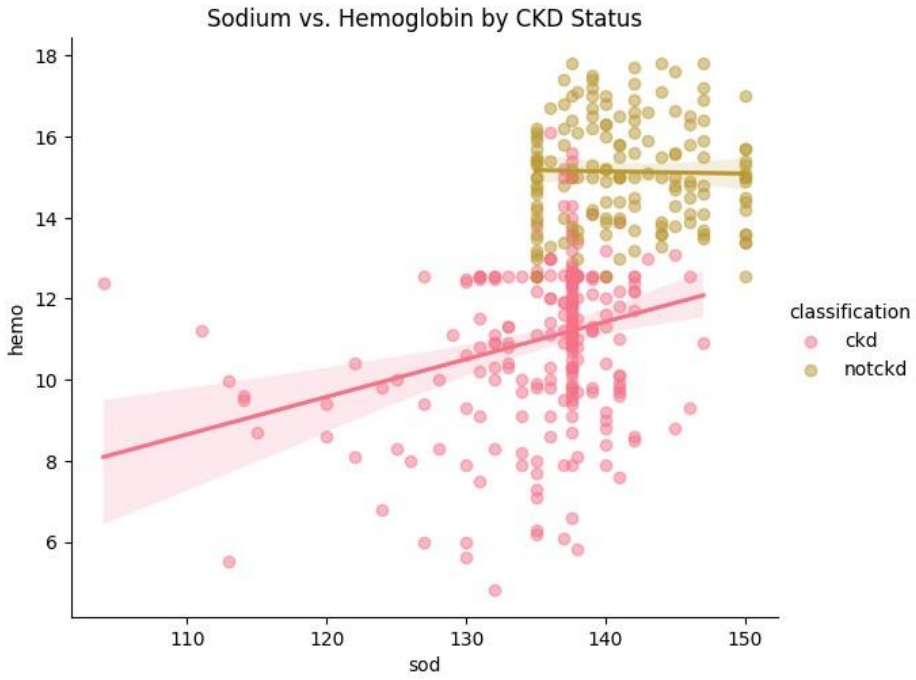

Blood Urea vs. RBC by CKD Status
rc
bu
classification
ckd
notckd


Serum Creatinine vs. Sodium by CKD Status
sod
sc
classification
ckd
notckd


Blood Urea vs. Sodium by CKD Status
sod
bu
classification
ckd
notckd

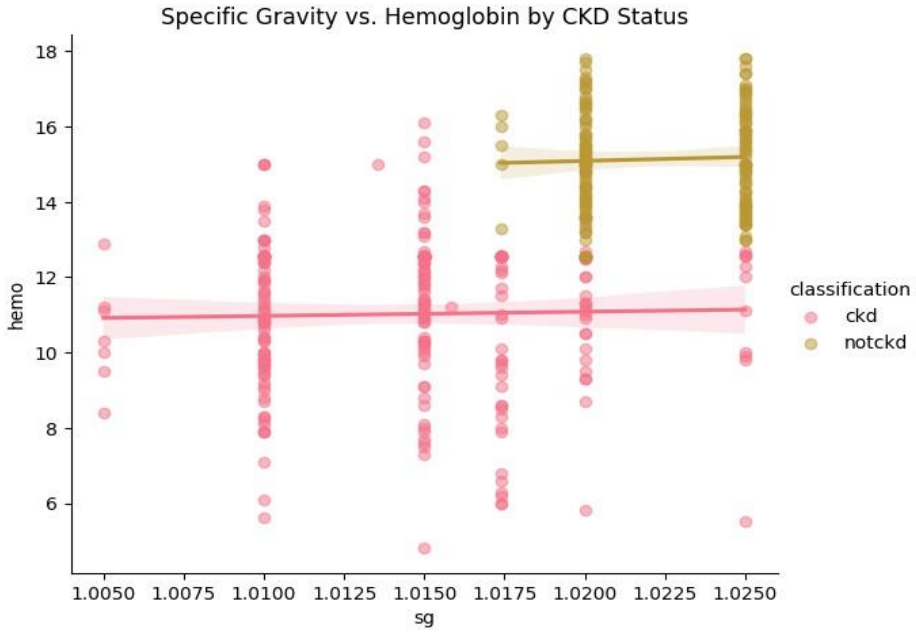
b. Sodium and Blood Health

- Sodium vs. Hemo/PCV/RC:
- Visualize how sodium imbalances relate to anemia.

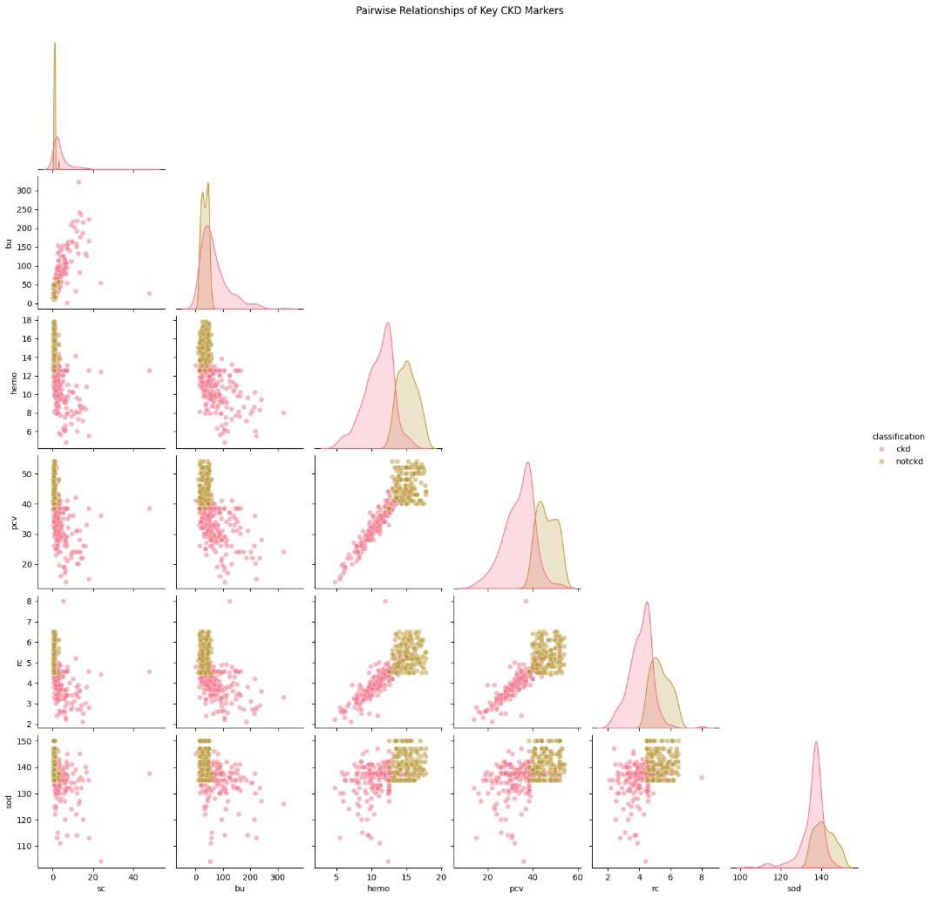


c. Specific Gravity and Blood Markers • SG vs. Hemo/PCV/RC:

- Higher SG may indicate better kidney function and blood health.



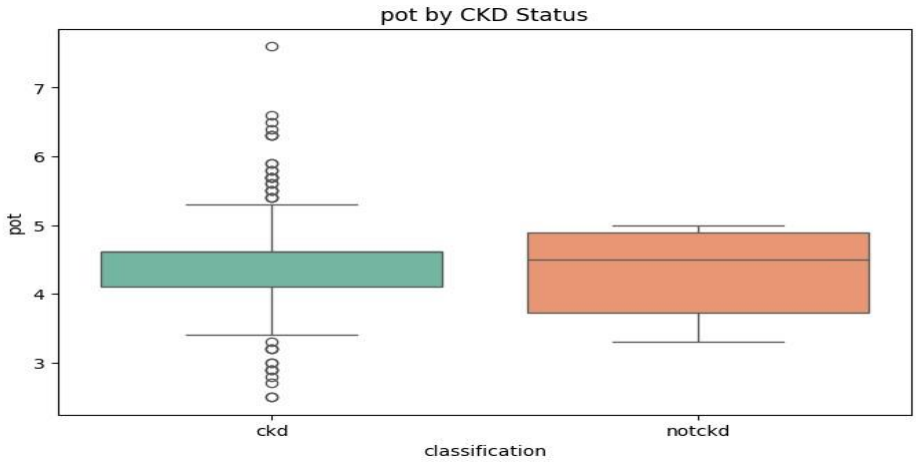
Multivariate Patterns: PairGrid/Pairplot with Hue:



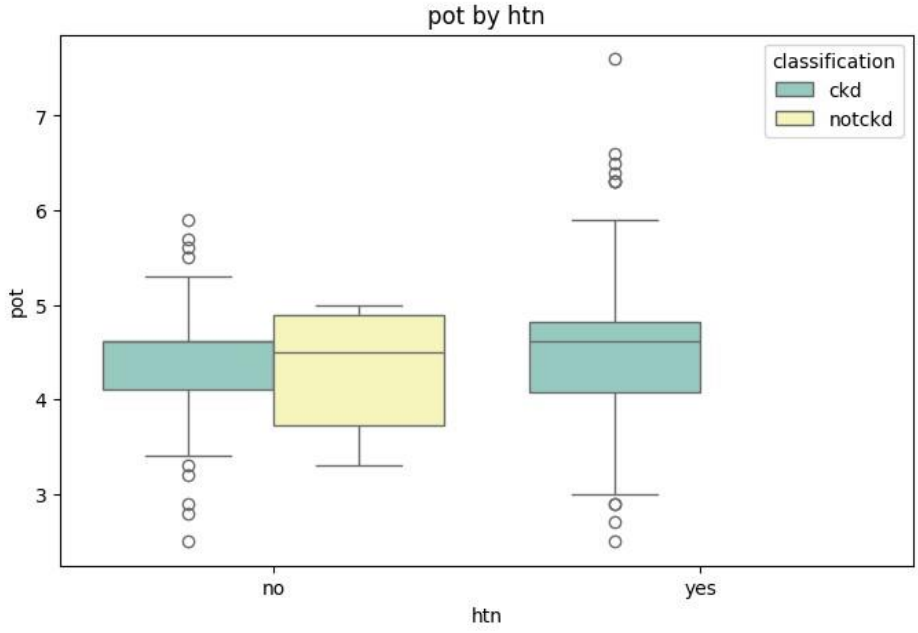
Explore Weak/Unexpected Correlations:

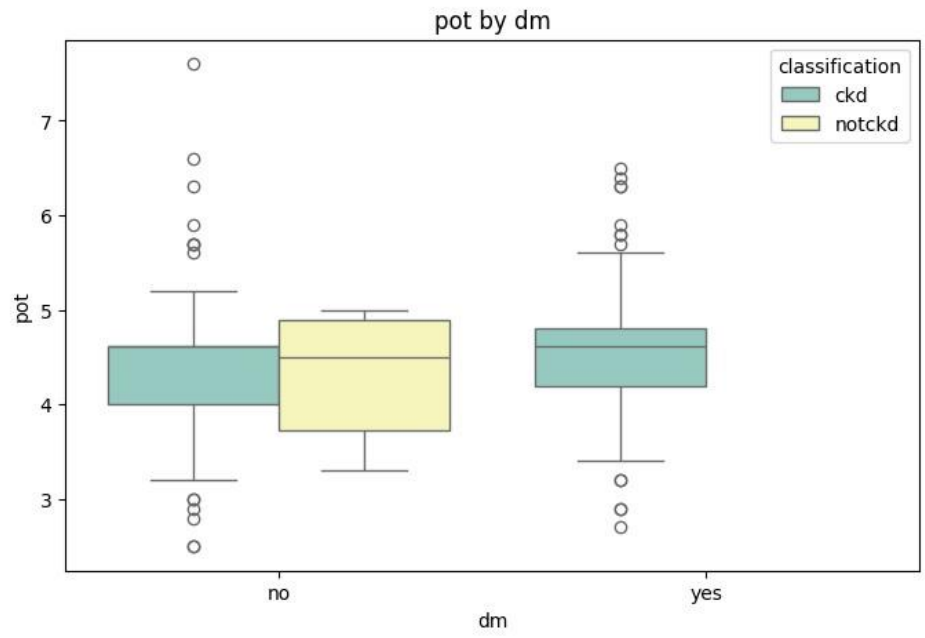
- Boxplots of Potassium, WBC, BP by CKD status and by other comorbidities (htn, dm, ane).

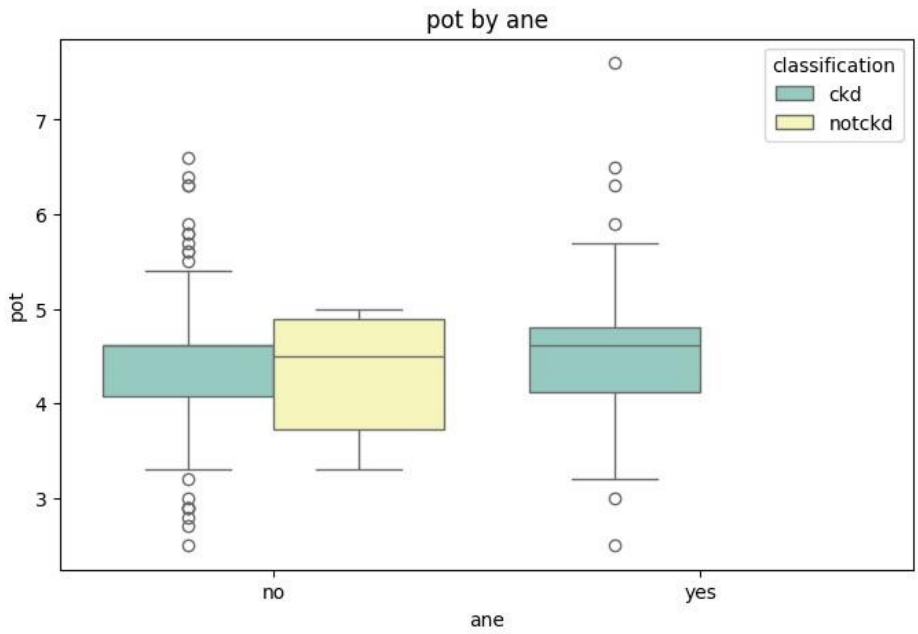
- This can reveal if these features are more important in certain

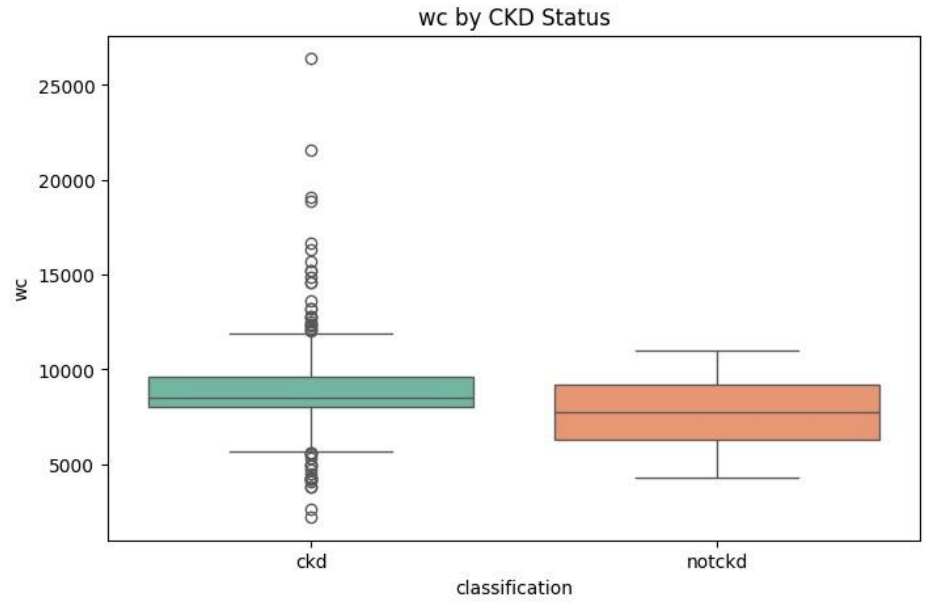


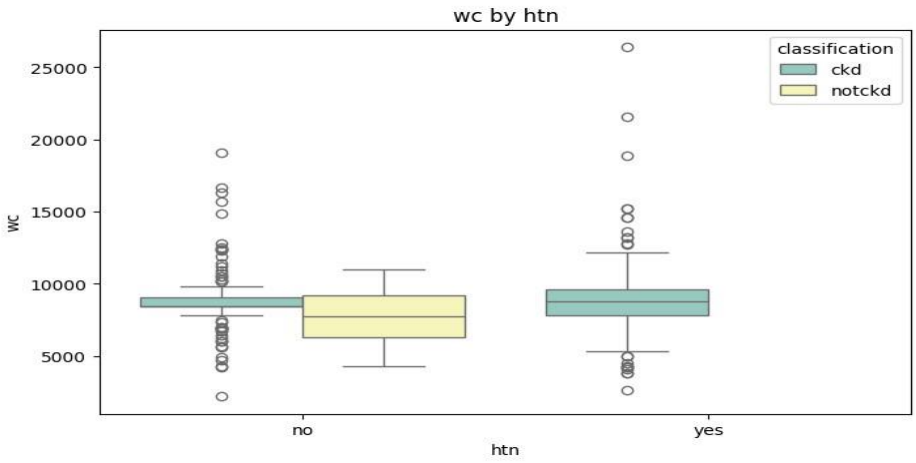
subgroups.

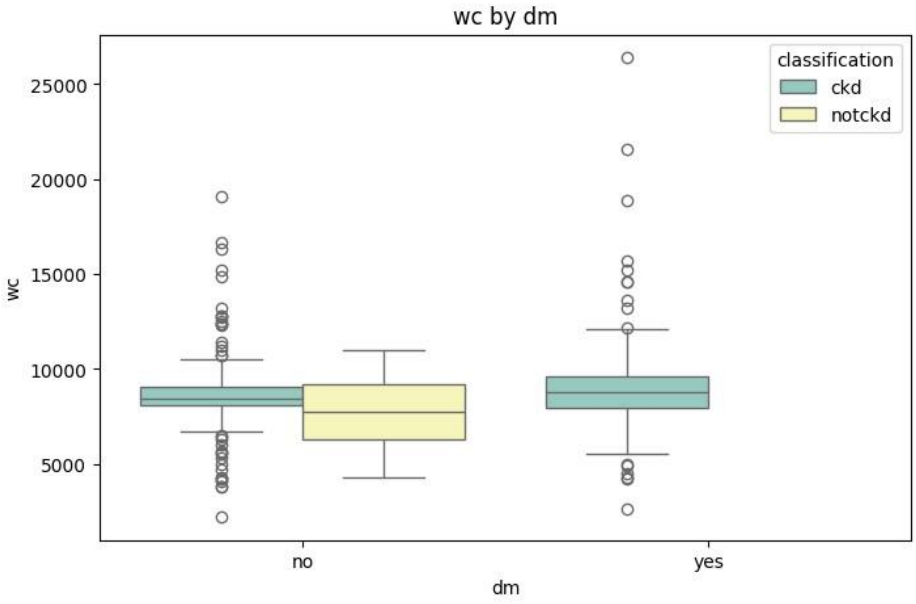

pot by htn
classification
ckd
notckd
pot
htn
no
yes
3
4
5
6
7

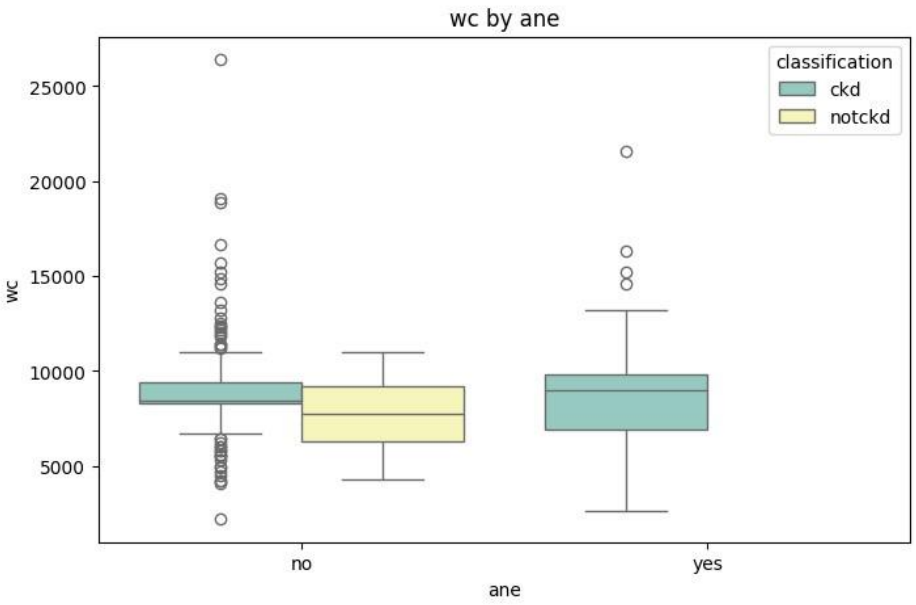

pot by dm
classification
ckd
notckd
pot
dm
no
yes
3
4
5
6
7

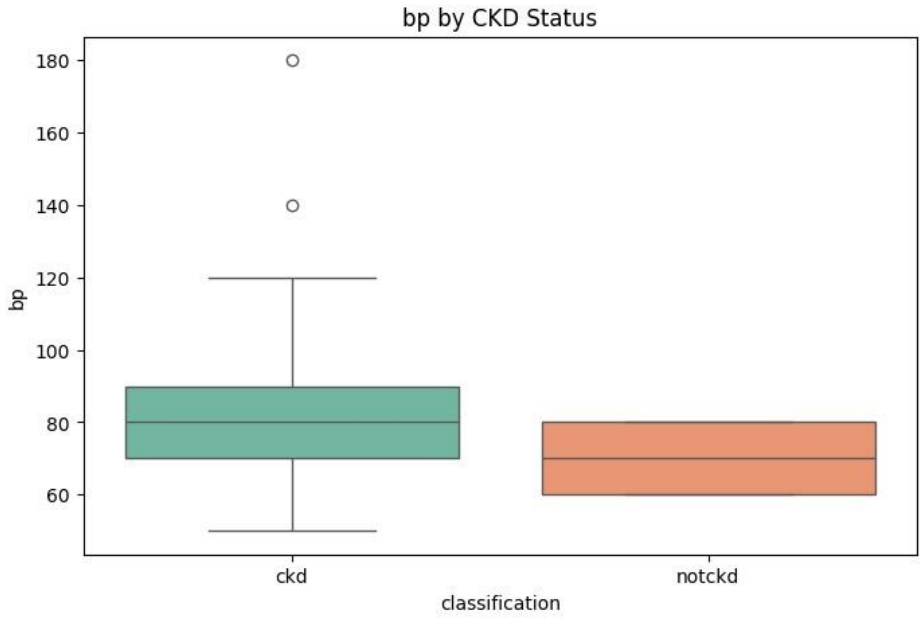

pot by ane
classification
ckd
notckd
pot
7
6
5
4
3
no
yes
ane

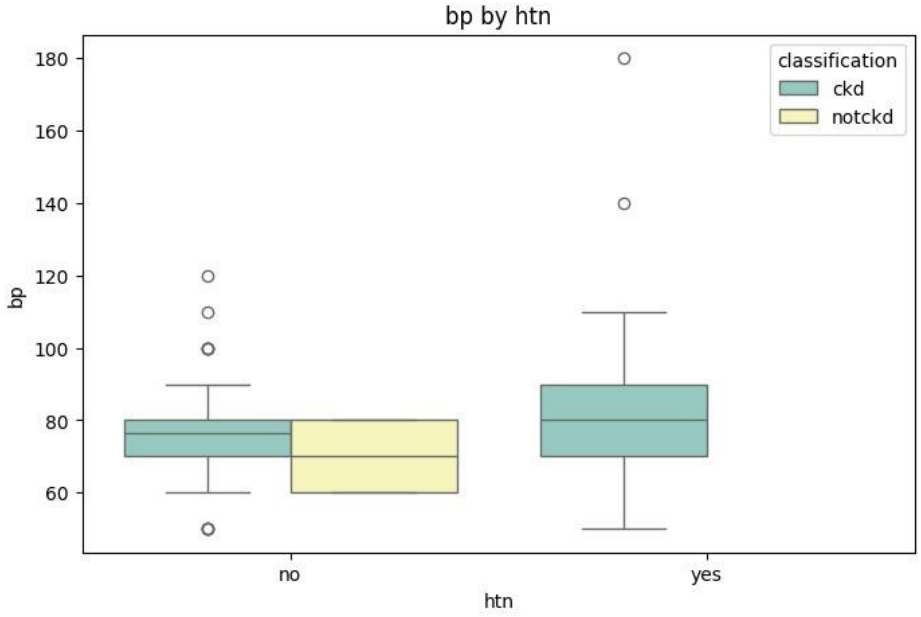

wc by CKD Status
wc
25000
20000
15000
10000
5000
ckd
notckd
classification

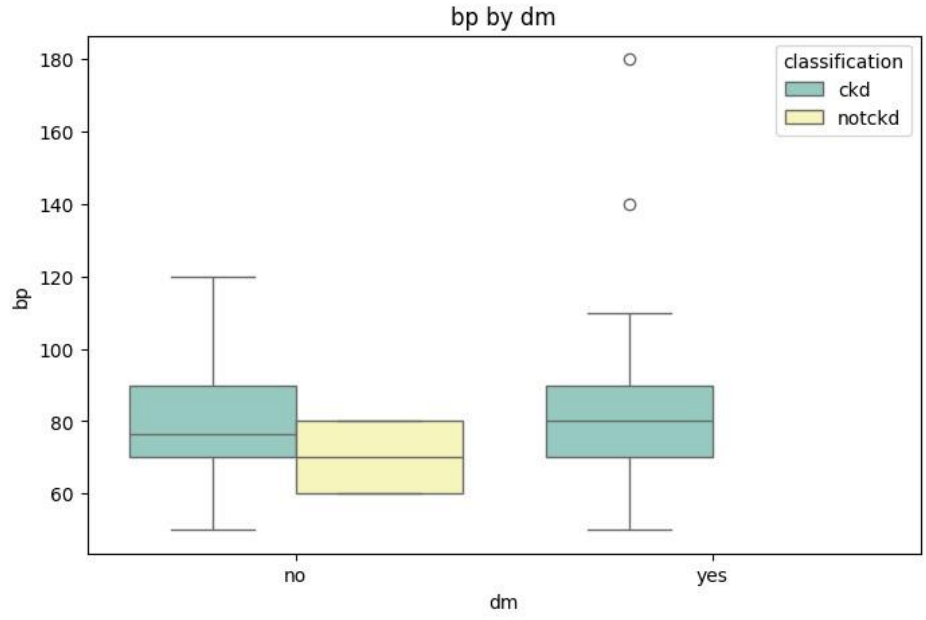

wc by htn
classification
ckd
notckd
25000
20000
15000
10000
5000
wc
no
yes
htn

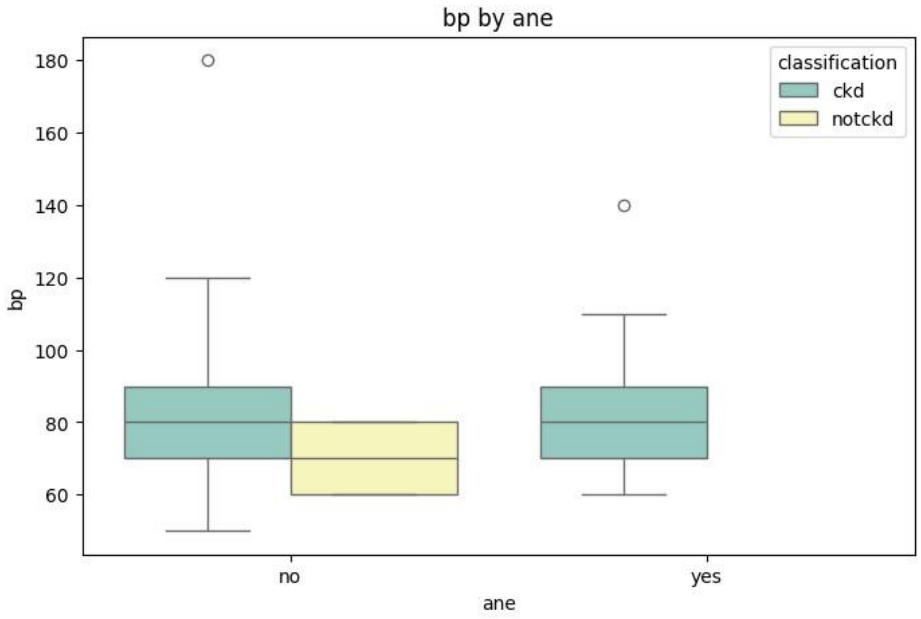

wc by dm
classification
ckd
notckd
25000
20000
15000
10000
5000
wc
no
yes
dm


wc by ane
classification
ckd
notckd
25000
20000
15000
10000
5000
wc
no
yes
ane


bp by CKD Status
180
160
140
120
100
80
60
bp
ckd
notckd
classification

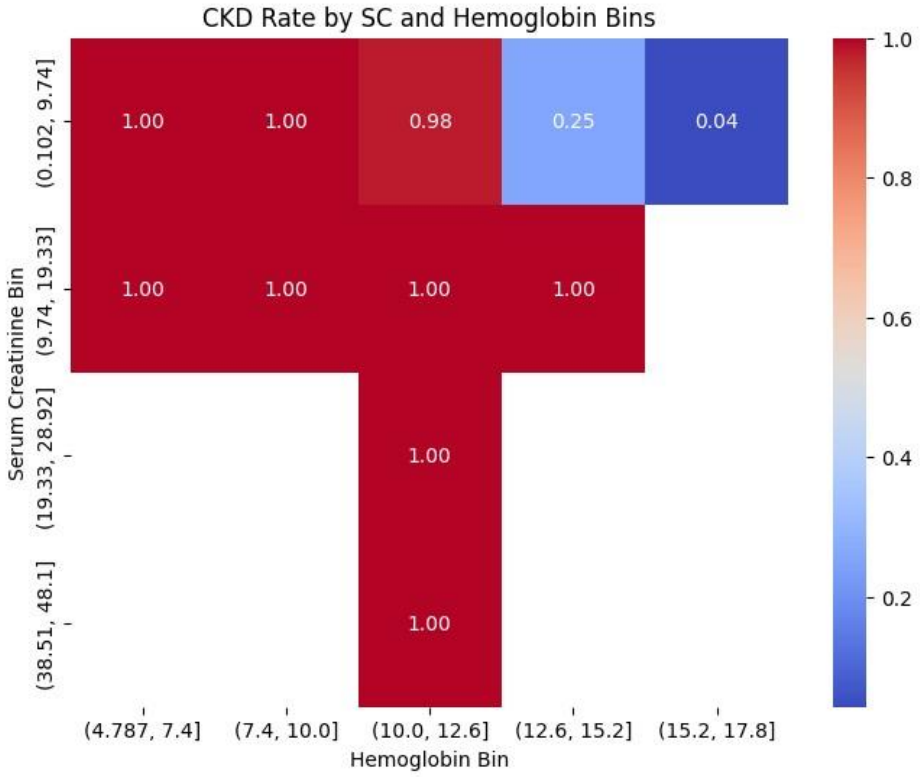

bp by htn
classification
ckd
notckd
bp
htn
no
yes
180
160
140
120
100
80
60

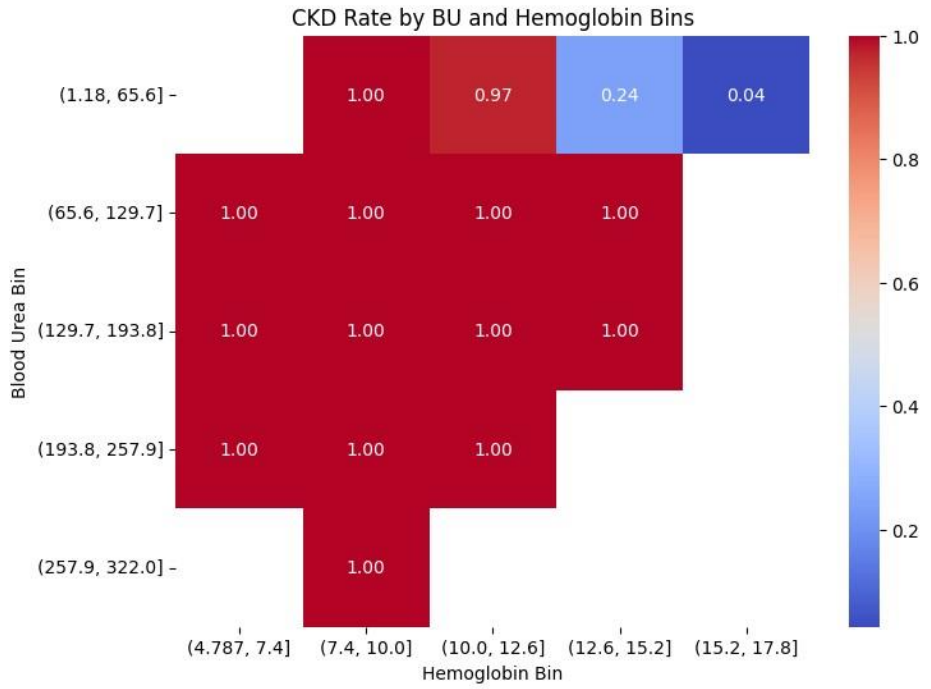

bp by dm
classification
ckd
notckd
bp
dm
no
yes
180
160
140
120
100
80
60



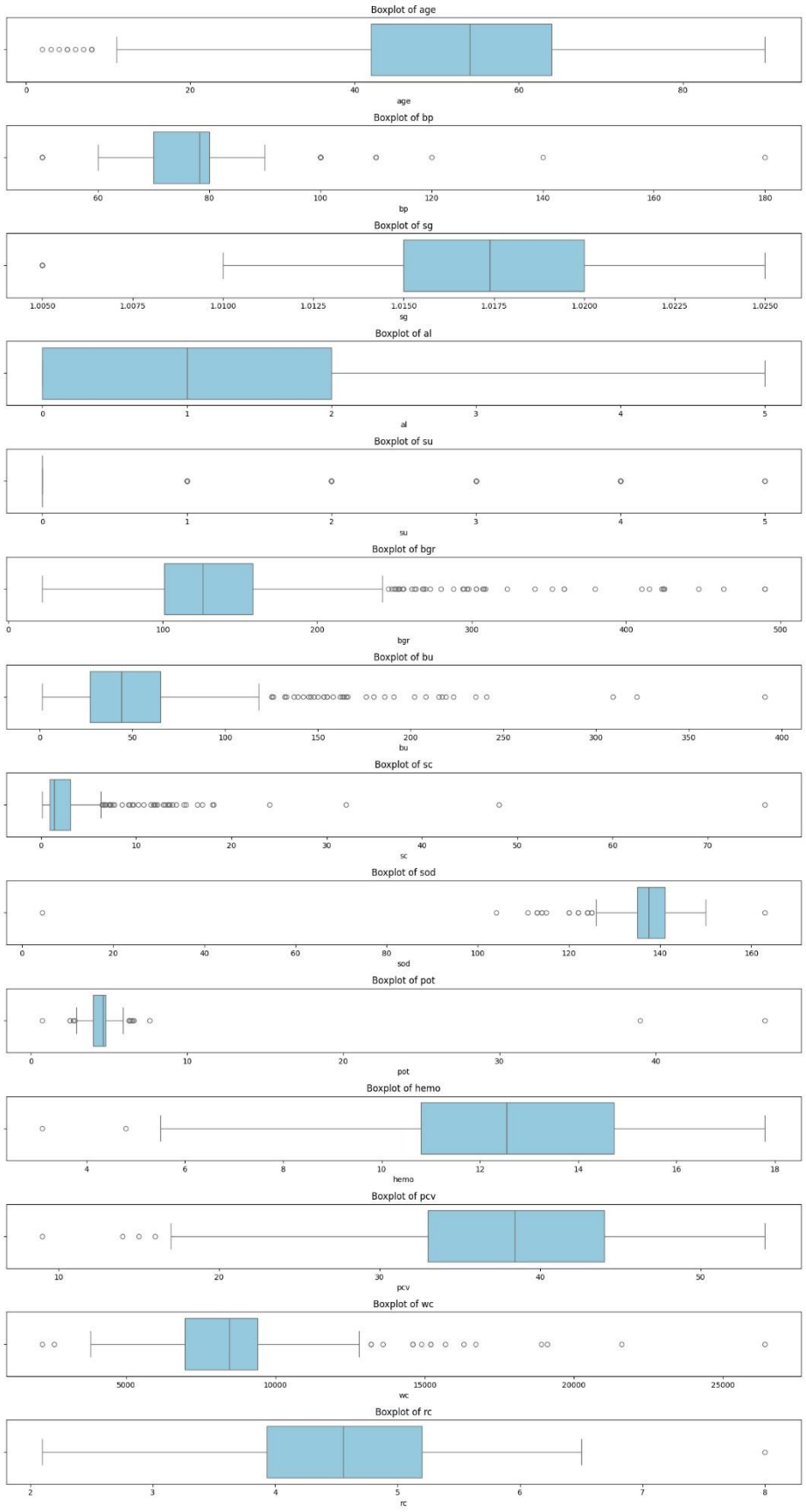
Interaction Plots:

- Interaction between two features and CKD status.
- Example: How does the relationship between SC/BU [Kidney Functioning Indicators] and Hemo differ by CKD status?



<table>
<tr>
<td></td>
<td> 
</td>
</tr>
<tr>
<td>Outliers and Anomalies</td>
<td>Outlier Analysis and Removal:

```
numerical_cols = df_clean.select_dtypes(include=['float64', 'int64']).columns.tolist()
numerical_cols = [col for col in numerical_cols if col not in ['id']]  # Exclude id

plt.figure(figsize=(15, 2 * len(numerical_cols)))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(len(numerical_cols), 1, i)
    sns.boxplot(x=df_clean[col], color='skyblue')
    plt.title(f'Boxplot of {col}')
    plt.tight_layout()
plt.show()
```
</td>
</tr>
</table>


Boxplot of age
age
Boxplot of bp
bp
Boxplot of sg
sg
Boxplot of al
al
Boxplot of su
su
Boxplot of bgr
bgr
Boxplot of bu
bu
Boxplot of sc
sc
Boxplot of sod
sod
Boxplot of pot
pot
Boxplot of hemo
hemo
Boxplot of pcv
pcv
Boxplot of wc
wc
Boxplot of rc
rc

Number of Outliers in Numerical Columns:

```
[ ] outlier_summary = {}
    for col in numerical_cols:
        Q1 = df_clean[col].quantile(0.25)
        Q3 = df_clean[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = df_clean[(df_clean[col] < lower_bound) | (df_clean[col] > upper_bound)][col]
        outlier_summary[col] = {
            'lower_bound': lower_bound,
            'upper_bound': upper_bound,
            'num_outliers': outliers.count(),
            'outlier_values': outliers.values
        }
        print(f"{col}: {outliers.count()} outliers")
```

```
age: 10 outliers
bp: 36 outliers
sg: 7 outliers
al: 0 outliers
su: 65 outliers
bgr: 43 outliers
bu: 39 outliers
sc: 44 outliers
sod: 18 outliers
pot: 15 outliers
hemo: 2 outliers
pcv: 4 outliers
wc: 17 outliers
rc: 1 outliers
```

Outlier Summary to Take Further Action:

```
for col in outlier_summary:
    print(f"{col}: {outlier_summary[col]['num_outliers']} outliers")
    print(f"Lower Bound: {outlier_summary[col]['lower_bound']}")
    print(f"Upper Bound: {outlier_summary[col]['upper_bound']}")
    print(f"Outlier Values: {outlier_summary[col]['outlier_values']}")
    print("---------------------------------")
```

```
age: 10 outliers
Lower Bound: 9.0
Upper Bound: 97.0
Outlier Values: [7. 5. 5. 8. 4. 8. 3. 8. 6. 2.]
---------------------------------
bp: 36 outliers
Lower Bound: 55.0
Upper Bound: 95.0
Outlier Values: [ 50. 100. 100. 100. 100. 100. 100. 100. 100. 100. 110. 100. 100. 140.
 180. 100. 100.  50. 100. 100. 100.  50.  50. 110. 100. 100. 100. 120.
 100. 100.  50. 100. 100. 100. 100. 110.]
---------------------------------
sg: 7 outliers
Lower Bound: 1.0074999999999998
Upper Bound: 1.0275000000000003
Outlier Values: [1.005 1.005 1.005 1.005 1.005 1.005 1.005]
---------------------------------
al: 0 outliers
Lower Bound: -3.0
Upper Bound: 5.0
Outlier Values: []
---------------------------------
su: 65 outliers
Lower Bound: 0.0
Upper Bound: 0.0
Outlier Values: [3 4 4 1 2 3 4 3 1 1 2 5 2 4 3 4 4 3 1 2 1 2 1 4 4 3 2 3 1 3 3 1 2 4 4 3 1
 1 3 2 2 4 2 5 2 1 4 2 1 1 1 2 2 2 4 1 3 5 2 2 3 1 2 3 1]
---------------------------------
```

```
bgr: 43 outliers
Lower Bound: 15.5
Upper Bound: 243.5
Outlier Values: [423. 410. 490. 380. 263. 264. 270. 246. 253. 425. 250. 360. 360. 415.
 251. 280. 295. 298. 297. 294. 323. 308. 268. 256. 288. 273. 424. 303.
 307. 447. 309. 261. 352. 252. 341. 255. 253. 248. 303. 490. 269. 463.
 424.]
--------------------------------
bu: 39 outliers
Lower Bound: -30.374999999999996
Upper Bound: 122.625
Outlier Values: [162.         148.         180.         163.         155.
 153.         202.         164.         155.         142.
 391.         139.         186.         217.         219.
 125.         125.         166.         208.         176.
 125.         125.91348805 145.         165.         322.
 235.         132.         146.         133.         153.
 137.         223.         158.         132.         150.
 191.         241.         215.         309.        ]
--------------------------------
sc: 44 outliers
Lower Bound: -2.3549614182191325
Upper Bound: 6.324935697031887
Outlier Values: [24.   7.2  9.6 76.   7.7  7.3 10.8  9.7  7.3  6.4 32.   6.7  6.7  8.5
 15.   6.5 10.2 11.5 12.2  9.2 13.8 16.9  7.1 18.  13.  48.1 14.2 16.4
  7.3  7.5  6.5 18.1 11.8  9.3  7.3  6.8 13.5 12.8 11.9  7.2 12.  13.4
 15.2 13.3]
--------------------------------
sod: 18 outliers
Lower Bound: 126.0
Upper Bound: 150.0
Outlier Values: [111.  104.  114.    4.5 125.  163.  122.  124.  115.  113.  125.  113.
 122.  124.  114.  120.  120.  124. ]
--------------------------------
 pot: 15 outliers
 Lower Bound: 2.8000000000000003
 Upper Bound: 6.0
 Outlier Values: [ 2.5         6.4         0.74393774  6.6         39.          7.6
  47.          6.3         2.5         2.8         2.7         6.5
   6.3         6.3         6.5       ]
 --------------------------------
 hemo: 2 outliers
 Lower Bound: 4.912500000000002
 Upper Bound: 20.612499999999997
 Outlier Values: [4.8 3.1]
 --------------------------------
 pcv: 4 outliers
 Lower Bound: 16.5
 Upper Bound: 60.5
 Outlier Values: [16. 14. 15.  9.]
 --------------------------------
 wc: 17 outliers
 Lower Bound: 3337.5
 Upper Bound: 13037.5
 Outlier Values: [18900. 21600. 14600. 13200. 13600. 14900. 15200. 16300. 13200. 15200.
  14600.  2200. 19100. 16700.  2600. 26400. 15700.]
 --------------------------------
 rc: 1 outliers
 Lower Bound: 2.0270219951450343
 Upper Bound: 7.10378680291298
 Outlier Values: [8.]
 --------------------------------
```

Outlier Handling:

We checked all the medical data ranges for all these numerical columns for which box plots were made and decided to remove only those that were clinically impossible. Other outliers were possible, but as a result

	of severe disease. Since the other records provided valuable medical information, we decided to keep them.

- sod (Sodium): ○ Removed rows where sod = 4.5 mEq/L (physiologically impossible).
  - Retained all other values, including extreme hyponatremia/hypernatremia, as these are possible in severe kidney disease.
- pot (Potassium): ○ Removed rows where pot ≥ 39 mmol/L (clinically impossible).
  - Retained all other values, including severe hypo/hyperkalemia, as these are possible in advanced disease.
- pcv (Packed Cell Volume): ○ Removed rows where pcv = 9% (clinically impossible). ○ Retained all other values, including severe anemia, as these are possible in advanced disease.
- All other columns: ○ Retained all outlier values, as they are medically possible, though some are rare or dangerous. This preserves the clinical diversity of the dataset for robust modeling.

```
[ ] # Count before removal
    removed_sod = np.sum(np.isclose(df_clean['sod'], 4.5, atol=0.01))
    removed_pot = np.sum(df_clean['pot'] >= 39)
    removed_pcv = np.sum(np.isclose(df_clean['pcv'], 9.0, atol=0.01))

    print(f"Rows to be removed for sod: {removed_sod}")
    print(f"Rows to be removed for pot: {removed_pot}")
    print(f"Rows to be removed for pcv: {removed_pcv}")

    # Now remove the impossible values
    df_clean = df_clean[~np.isclose(df_clean['sod'], 4.5, atol=0.01) | df_clean['sod'].isnull()]
    df_clean = df_clean[(df_clean['pot'].isnull()) | (df_clean['pot'] < 39)]
    df_clean = df_clean[~np.isclose(df_clean['pcv'], 9.0, atol=0.01)]
```

```
Rows to be removed for sod: 1
Rows to be removed for pot: 2
Rows to be removed for pcv: 1
```

Data Preprocessing Code Screenshots

Loading Data

```
df = pd.read_csv('chronickidneydisease.csv')
df.head()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	ckd

5 rows × 26 columns

Handling Missing Data	Cleaning all Numerical Columns (\t ' ' ? and Other Issues) And Changing Incorrect Datatypes

```
# Function to clean numerical columns
def clean_numerical_column(series):
    # Remove tab characters and convert to string
    series = series.astype(str).str.replace('\t', '')

    # Replace '?' with NaN
    series = series.replace('?', np.nan)

    # Convert to numeric, coercing errors to NaN, if in string format
    series = pd.to_numeric(series, errors='coerce')

    return series

# Clean the numerical columns stored as objects
numerical_columns_to_clean = ['pcv', 'wc', 'rc']
for col in numerical_columns_to_clean:
    df_clean[col] = clean_numerical_column(df_clean[col])

    # Print summary after cleaning
    print(f"\nColumn: {col}")
    print(f"Data type after cleaning: {df_clean[col].dtype}")
    print(f"Number of missing values: {df_clean[col].isnull().sum()}")
    print(f"Value range: {df_clean[col].min()} to {df_clean[col].max()}")
    print(f"Mean: {df_clean[col].mean():.2f}")
    print(f"Median: {df_clean[col].median():.2f}")
    print("---------------------------------")
```

```
Column: pcv
Data type after cleaning: float64
Number of missing values: 71
Value range: 9.0 to 54.0
Mean: 38.88
Median: 40.00
---------------------------------

Column: wc
Data type after cleaning: float64
Number of missing values: 106
Value range: 2200.0 to 26400.0
Mean: 8406.12
Median: 8000.00
---------------------------------

Column: rc
Data type after cleaning: float64
Number of missing values: 131
Value range: 2.1 to 8.0
Mean: 4.71
Median: 4.80
---------------------------------
```

Cleaning Categorical Columns:

```
unclean_categorical_columns=['classification','cad','dm']

def clean_categorical_column(series):
    series = series.astype(str).str.replace('\t','')
    series = series.astype(str).str.replace(' ', '')
    series = series.replace('?', np.nan)
    series= series.replace('nan',np.nan)
    return series

for column in unclean_categorical_columns:
    df_clean[column]=clean_categorical_column(df_clean[column])

    print(f"Column: {column}")
    print(f"Unique Values:\n {df_clean[column].unique()}\n")
    print(f"Value Counts:\n {df_clean[column].value_counts()}\n")
    print(f"Missing Values:\n {df_clean[column].isnull().sum()}\n")
    print(f"Data Type:\n {df_clean[column].dtype}\n")
    print(f"Description:\n {df_clean[column].describe()}\n")
    print("--------------------------------\n\n")
```

```
Column: classification
Unique Values:
 ['ckd' 'notckd']

Value Counts:
 classification
ckd       250
notckd    150
Name: count, dtype: int64

Missing Values:
 0

Data Type:
 object

Description:
 count     400
unique      2
top       ckd
freq      250
Name: classification, dtype: object

--------------------------------
```

```
Column: cad
Unique Values:
 ['no' 'yes' nan]

Value Counts:
 cad
no     364
yes     34
Name: count, dtype: int64

Missing Values:
 2

Data Type:
 object

Description:
 count     398
unique      2
top        no
freq      364
Name: cad, dtype: object

--------------------------------

Column: dm
Unique Values:
 ['yes' 'no' nan]

Value Counts:
 dm
no     261
yes    137
Name: count, dtype: int64

Missing Values:
 2

Data Type:
 object

Description:
 count     398
unique      2
top        no
freq      261
Name: dm, dtype: object

--------------------------------
```

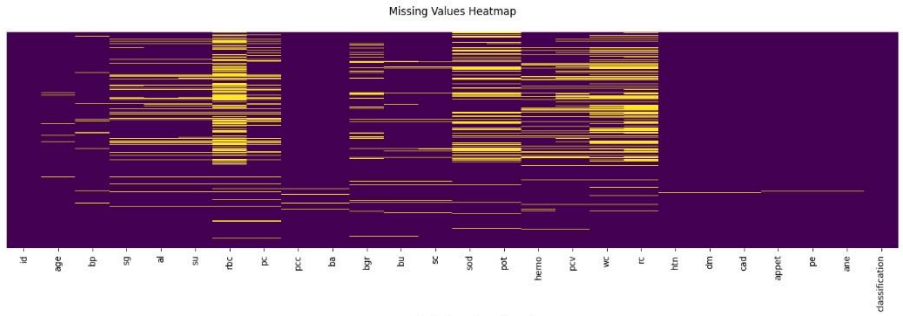
```
# Set the style for better visualization
sns.set_palette("husl")
```

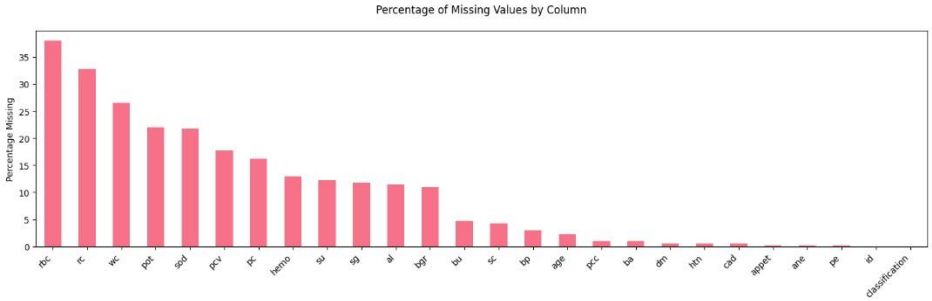
Missing Pattern Analysis for Integrity of Data:

```
# 1. Create a missing value heatmap
def plot_missing_heatmap(df):
    # Calculate missing values percentage for each column
    missing_percentage = (df.isnull().sum() / len(df)) * 100

    # Create a figure with two subplots
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 10))

    # Plot 1: Missing value heatmap
    sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis', ax=ax1)
    ax1.set_title('Missing Values Heatmap', pad=20)

    # Plot 2: Missing values percentage bar plot
    missing_percentage.sort_values(ascending=False).plot(kind='bar', ax=ax2)
    ax2.set_title('Percentage of Missing Values by Column', pad=20)
    ax2.set_ylabel('Percentage Missing')
    plt.xticks(rotation=45, ha='right')

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

```
    # Print detailed missing value statistics
    print("\nDetailed Missing Value Statistics:")
    print("-" * 50)
    for col in df.columns:
        missing = df[col].isnull().sum()
        percentage = (missing / len(df)) * 100
        print(f"{col}:")
        print(f"  Missing values: {missing} ({percentage:.2f}%)")
        if df[col].dtype == 'object':
            print(f"  Most common value: {df[col].value_counts().index[0] if not df[col].value_counts().empty else 'N/A'}")
        else:
            print(f"  Mean: {df[col].mean():.2f}")
            print(f"  Median: {df[col].median():.2f}")
        print("-" * 50)

# 2. Analyze patterns in missing values
def analyze_missing_patterns(df):
    # Create a correlation matrix of missing values
    missing_corr = df.isnull().corr()

    # Plot the correlation heatmap with adjusted settings
    plt.figure(figsize=(15, 12))  # Increased figure size
    sns.heatmap(missing_corr,
                annot=True,
                cmap='coolwarm',
                center=0,
                annot_kws={'size': 8},  # Smaller annotation font
                fmt='.2f')  # Format to 2 decimal places

    plt.title('Correlation of Missing Values Between Columns', pad=20)
    plt.xticks(rotation=45, ha='right', fontsize=8)  # Rotated and smaller x-tick labels
    plt.yticks(fontsize=8)  # Smaller y-tick labels
    plt.tight_layout()
    plt.show()

    # Analyze if missing values are related to the target variable
    if 'classification' in df.columns:
        print("\nMissing Values by Target Variable:")
        print("-" * 70)  # Wider separator
        for col in df.columns:
            if df[col].isnull().any():
                missing_by_target = df.groupby('classification')[col].apply(lambda x: x.isnull().mean())
                print(f"\n{col:15}")  # Fixed width for column names
                print("-" * 30)
                for target, value in missing_by_target.items():
                    print(f"{target:10}: {value:.2%}")  # Format as percentage
                print("-" * 70)
```


```
# Run the analysis
print("Step 1: Visualizing Missing Values")
plot_missing_heatmap(df_clean)

print("\nStep 2: Analyzing Missing Value Patterns")
analyze_missing_patterns(df_clean)
```

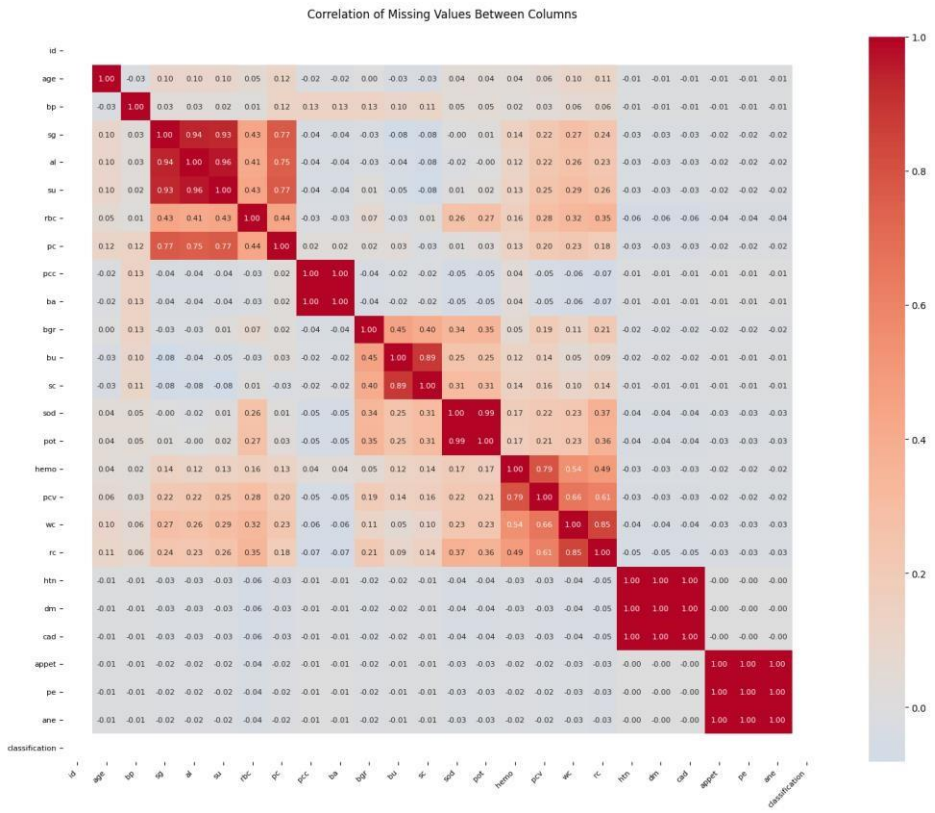





```
Detailed Missing Value Statistics:
--------------------------------------------------
id:
  Missing values: 0 (0.00%)
  Mean: 199.50
  Median: 199.50
--------------------------------------------------
age:
  Missing values: 9 (2.25%)
  Mean: 51.48
  Median: 55.00
--------------------------------------------------
bp:
  Missing values: 12 (3.00%)
  Mean: 76.47
  Median: 80.00
--------------------------------------------------
sg:
  Missing values: 47 (11.75%)
  Mean: 1.02
  Median: 1.02
--------------------------------------------------
al:
  Missing values: 46 (11.50%)
  Mean: 1.02
  Median: 0.00
--------------------------------------------------
```

```
su:
  Missing values: 49 (12.25%)
  Mean: 0.45
  Median: 0.00
------------------------------------------------
rbc:
  Missing values: 152 (38.00%)
  Most common value: normal
------------------------------------------------
pc:
  Missing values: 65 (16.25%)
  Most common value: normal
------------------------------------------------
pcc:
  Missing values: 4 (1.00%)
  Most common value: notpresent
------------------------------------------------
ba:
  Missing values: 4 (1.00%)
  Most common value: notpresent
------------------------------------------------
bgr:
  Missing values: 44 (11.00%)
  Mean: 148.04
  Median: 121.00
------------------------------------------------
------------------------------------------------
bu:
  Missing values: 19 (4.75%)
  Mean: 57.43
  Median: 42.00
------------------------------------------------
sc:
  Missing values: 17 (4.25%)
  Mean: 3.07
  Median: 1.30
------------------------------------------------
sod:
  Missing values: 87 (21.75%)
  Mean: 137.53
  Median: 138.00
------------------------------------------------
pot:
  Missing values: 88 (22.00%)
  Mean: 4.63
  Median: 4.40
------------------------------------------------
hemo:
  Missing values: 52 (13.00%)
  Mean: 12.53
  Median: 12.65
------------------------------------------------
pcv:
  Missing values: 71 (17.75%)
  Mean: 38.88
  Median: 40.00
------------------------------------------------
```

```
wc:
  Missing values: 106 (26.50%)
  Mean: 8406.12
  Median: 8000.00
--------------------------------------------------
rc:
  Missing values: 131 (32.75%)
  Mean: 4.71
  Median: 4.80
--------------------------------------------------
htn:
  Missing values: 2 (0.50%)
  Most common value: no
--------------------------------------------------
dm:
  Missing values: 2 (0.50%)
  Most common value: no
--------------------------------------------------
cad:
  Missing values: 2 (0.50%)
  Most common value: no
--------------------------------------------------
appet:
  Missing values: 1 (0.25%)
  Most common value: good
--------------------------------------------------
pe:
  Missing values: 1 (0.25%)
  Most common value: no
--------------------------------------------------
pe:
  Missing values: 1 (0.25%)
  Most common value: no
--------------------------------------------------
ane:
  Missing values: 1 (0.25%)
  Most common value: no
--------------------------------------------------
classification:
  Missing values: 0 (0.00%)
  Most common value: ckd
--------------------------------------------------
```




```
Missing Values by Target Variable:
--------------------------------------------------------------------------------

age
--------------------------------
ckd       : 3.20%
notckd    : 0.67%
--------------------------------------------------------------------------------

bp
--------------------------------
ckd       : 4.00%
notckd    : 1.33%
--------------------------------------------------------------------------------

sg
--------------------------------
ckd       : 16.80%
notckd    : 3.33%
--------------------------------------------------------------------------------

al
--------------------------------
ckd       : 16.40%
notckd    : 3.33%
--------------------------------------------------------------------------------
```

```
su
------------------------------
ckd       : 17.60%
notckd    : 3.33%
--------------------------------------------------------------------------------

rbc
------------------------------
ckd       : 57.20%
notckd    : 6.00%
--------------------------------------------------------------------------------

pc
------------------------------
ckd       : 22.40%
notckd    : 6.00%
--------------------------------------------------------------------------------

pcc
------------------------------
ckd       : 0.00%
notckd    : 2.67%
--------------------------------------------------------------------------------

ba
------------------------------
ckd       : 0.00%
notckd    : 2.67%
--------------------------------------------------------------------------------

bgr
------------------------------
ckd       : 15.20%
notckd    : 4.00%
--------------------------------------------------------------------------------

bu
------------------------------
ckd       : 5.20%
notckd    : 4.00%
--------------------------------------------------------------------------------

sc
------------------------------
ckd       : 4.80%
notckd    : 3.33%
--------------------------------------------------------------------------------

sod
------------------------------
ckd       : 32.80%
notckd    : 3.33%
--------------------------------------------------------------------------------

pot
------------------------------
ckd       : 33.20%
notckd    : 3.33%
--------------------------------------------------------------------------------
```

```
hemo
------------------------------
ckd        : 18.40%
notckd     : 4.00%
------------------------------------------------------------------------

pcv
------------------------------
ckd        : 26.80%
notckd     : 2.67%
------------------------------------------------------------------------

wc
------------------------------
ckd        : 39.60%
notckd     : 4.67%
------------------------------------------------------------------------

rc
------------------------------
ckd        : 49.60%
notckd     : 4.67%
------------------------------------------------------------------------

htn
------------------------------
ckd        : 0.00%
notckd     : 1.33%
------------------------------------------------------------------------

dm
------------------------------
ckd        : 0.00%
notckd     : 1.33%
------------------------------------------------------------------------

cad
------------------------------
ckd        : 0.00%
notckd     : 1.33%
------------------------------------------------------------------------

appet
------------------------------
ckd        : 0.00%
notckd     : 0.67%
------------------------------------------------------------------------

pe
------------------------------
ckd        : 0.00%
notckd     : 0.67%
------------------------------------------------------------------------

ane
------------------------------
ckd        : 0.00%
notckd     : 0.67%
------------------------------------------------------------------------
```

Way of Handling Missing Values:

- Low Missing (<5%)>:
  - Categorical
    - Filled with mode
  - Numerical

- Filled with mean

- Too Many (>30%): ○ Categorical
  - Make separate category called missing
  - ○ Numerical
    - None found
- Intermediate Amount (5-30%): ○ Mice Imputer using default imputer model

```
low_missing_cats = ['pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane']
for col in low_missing_cats:
    mode_val = df_clean[col].mode()[0]
    df_clean[col].fillna(mode_val, inplace=True)
```

```
[ ]  df_clean['rbc'] = df_clean['rbc'].fillna('missing')
     df_clean['pc'] = df_clean['pc'].fillna('missing')
```

```
[ ]  from sklearn.experimental import enable_iterative_imputer  # noqa
     from sklearn.impute import IterativeImputer
     from sklearn.preprocessing import StandardScaler

     def mice_impute_group(df, columns, estimator=None, max_iter=10, random_state=0):
         scaler = StandardScaler()
         scaled = scaler.fit_transform(df[columns])
         imputer = IterativeImputer(estimator=estimator, max_iter=max_iter, random_state=random_state)
         imputed = imputer.fit_transform(scaled)
         df[columns] = scaler.inverse_transform(imputed)
         return df
```

- Mice Imputer was chosen because it can impute columns together if they tend to be missing together as observed earlier through the missing value heatmap and correlation matrix based on patterns in the data, instead of imputing randomly ○ Relevant for medical data ○ Ensures integrity is maintained

```
[ ]  # Impute each group
     df_clean = mice_impute_group(df_clean, ['sod', 'pot'])
     df_clean = mice_impute_group(df_clean, ['sg', 'al', 'su'])
     df_clean = mice_impute_group(df_clean, ['hemo', 'pcv', 'wc', 'rc'])
     df_clean = mice_impute_group(df_clean, ['bu', 'sc', 'bgr'])  # bgr included for completeness
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/impute/_iterative.py:895: ConvergenceWarning:
  warnings.warn(
```

```
[ ]  df_clean['age'].fillna(df_clean['age'].mean(), inplace=True)
     df_clean['bp'].fillna(df_clean['bp'].mean(), inplace=True)
```

```
[ ] print(df_clean.isnull().sum())
```

```
id                0
age               0
bp                0
sg                0
al                0
su                0
rbc               0
pc                0
pcc               0
ba                0
bgr               0
bu                0
sc                0
sod               0
pot               0
hemo              0
pcv               0
wc                0
rc                0
htn               0
dm                0
cad               0
appet             0
pe                0
ane               0
classification    0
dtype: int64
```

'al' and 'su' Are Categorical Columns With Oridinal Categories From 0-5

```
] for col in ['al', 'su']:
      df_clean[col] = df_clean[col].round().astype(int)
      df_clean[col] = df_clean[col].clip(lower=0, upper=5)
```

Feature Engineering	Feature Engineering for CKD: 1. eGFR Calculation (Estimated Glomerular Filtration Rate) • eGFR is a standard kidney function score, calculated from serum

creatinine, age, sex, and sometimes race.

- Since you don't have sex/race, use the simplified MDRD formula (for adults):
  - eGFR=186×(sc)^(−1.154)×(age)^(−0.203)

```
# Avoid division by zero or negative values
df_clean['eGFR'] = 186 * (df_clean['sc'].clip(lower=0.01))**(-1.154) * (df_clean['age'].clip(lower=1))**(-0.203)
```

2. Comorb_Score (Level of Comorbid Conditions in One's Body)
   - 0 (none) to 3 (all conditions)
   - Each "yes" = 1 point.

```
df_clean['comorb_score'] = (
    (df_clean['htn'] == 'yes').astype(int) +
    (df_clean['dm'] == 'yes').astype(int) +
    (df_clean['cad'] == 'yes').astype(int)
)
```

3. Anemia Severity Score
   - Combine hemo, pcv, rc into a z-score-based severity index.
   - Interpretation: Higher value = more severe anemia.

```
from scipy.stats import zscore

# Calculate z-scores (lower = more severe anemia)
df_clean['anemia_severity'] = -(
    zscore(df_clean['hemo']) +
    zscore(df_clean['pcv']) +
    zscore(df_clean['rc'])
)
```

4. Kidney Function Score
   - Interpretation: Higher value = worse kidney function.

```
df_clean['kidney_func_score'] = (
    zscore(df_clean['bu']) +
    zscore(df_clean['sc']) -
    zscore(df_clean['sod'])
)
```

5. Symptom Severity Score
   - Combine appet, pe, ane (each "poor"/"yes" = 1 point).
   - Interpretation: Higher score = more severe symptoms.

```
df_clean['symptom_severity'] = (
    (df_clean['appet'] == 'poor').astype(int) +
    (df_clean['pe'] == 'yes').astype(int) +
    (df_clean['ane'] == 'yes').astype(int)
)
```

Data Transformation

Transformation and Scaling:
Data Transformation Steps:

- Log-transformed highly skewed variables: bgr, bu, sc, wc, eGFR.
- Standard scaled all numerical features.
- One-hot encoded all categorical variables.
- All transformations were fit on the training set and applied to the test set to prevent data leakage.

Ensuring None That Require Log Transforms Have 0 or -ve Values:

```
cols_to_check = ['bgr', 'bu', 'sc', 'wc', 'eGFR']

for col in cols_to_check:
    if col in df_clean.columns:
        num_zeros = (df_clean[col] == 0).sum()
        num_neg = (df_clean[col] < 0).sum()
        print(f"{col}: {num_zeros} zeros, {num_neg} negatives")
        print(f"Min value: {df_clean[col].min()}")
        print(f"Type: {df_clean[col].dtype}\n")
```

```
bgr: 0 zeros, 0 negatives
Min value: 22.0
Type: float64

bu: 0 zeros, 0 negatives
Min value: 1.5
Type: float64

sc: 0 zeros, 0 negatives
Min value: 0.14948944464248815
Type: float64

wc: 0 zeros, 0 negatives
Min value: 2200.0
Type: float64

eGFR: 0 zeros, 0 negatives
Min value: 0.9016358277259284
Type: float64
```

Save Processed Data

```
from sklearn.model_selection import train_test_split

# Exclude target and categorical columns from features
target = 'classification'
feature_cols = [col for col in df_clean.columns if col not in [target, 'sc_bin', 'hemo_bin', 'bu_bin']]  # Exclude binned cols and target

X = df_clean[feature_cols]
y = df_clean[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
print(X.columns.tolist())
```

```
['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc',
'ba', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv',
'wc', 'rc', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane',
'eGFR', 'comorb_score', 'anemia_severity',
'kidney_func_score', 'symptom_severity']
```

```
print(y)
```

```
0         ckd
1         ckd
2         ckd
3         ckd
4         ckd
        ...
395    notckd
396    notckd
397    notckd
398    notckd
399    notckd
Name: classification, Length: 396, dtype: category
Categories (2, object): ['ckd', 'notckd']
```