Univariate Analysis of CKD Dataset

Introduction

Objective: Perform **Univariate Analysis** of CKD dataset

Focus: Explore **numerical** and **categorical** variables individually.

Scope:

1. Handle missing values & outliers.
2. Summarize **numerical variables** (central tendency, dispersion, skewness, kurtosis).
3. Summarize **categorical variables** (frequency, relative frequency, distribution).
4. Provide visualizations & interpretations.

Data Preparation:

Source: CKD dataset from UCI ML repository.
Cleaning:

1. Replaced "?" with NaN.
2. Converted columns to numeric where applicable.
3. Handled missing values using **median imputation**.

```python
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

# Load dataset
dataset = pd.read_csv("kidney_disease.csv")

# Rename columns properly
dataset.columns = [
    'id','age','blood_pressure','specific_gravity','albumin','sugar',
    'red_blood_cells','pus_cell','pus_cell_clumps','bacteria',
    'blood_glucose_random','blood_urea','serum_creatinine','sodium',
    'potassium','haemoglobin','packed_cell_volume','white_blood_cell_count',
    'red_blood_cell_count','hypertension','diabetes_mellitus',
    'coronary_artery_disease','appetite','peda_edema','aanemia','classification'
]

# Select numerical columns of interest
num_cols = ['age','blood_pressure','blood_glucose_random','blood_urea',
            'serum_creatinine','sodium','potassium','haemoglobin']
df = dataset[num_cols].replace("?", pd.NA)

# Convert to numeric
for c in df.columns:
    df[c] = pd.to_numeric(df[c], errors="coerce")

# Impute missing values with median
imp = SimpleImputer(strategy="median")
df_imputed = pd.DataFrame(imp.fit_transform(df), columns=df.columns, index=df.index)
df_imputed
```

	age	blood_pressure	blood_glucose_random	blood_urea	serum_creatinine	sodium	potassium	haemoglobin
0	48.0	80.0	121.0	36.0	1.2	138.0	4.4	15.4
1	7.0	50.0	121.0	18.0	0.8	138.0	4.4	11.3
2	62.0	80.0	423.0	53.0	1.8	138.0	4.4	9.6
3	48.0	70.0	117.0	56.0	3.8	111.0	2.5	11.2
4	51.0	80.0	106.0	26.0	1.4	138.0	4.4	11.6
...	...	...	...	...	...	...	...	...
395	55.0	80.0	140.0	49.0	0.5	150.0	4.9	15.7
396	42.0	70.0	75.0	31.0	1.2	141.0	3.5	16.5
397	12.0	80.0	100.0	26.0	0.6	137.0	4.4	15.8
398	17.0	60.0	114.0	50.0	1.0	135.0	4.9	14.2
399	58.0	80.0	131.0	18.0	1.1	141.0	3.5	15.8

400 rows × 8 columns

Numerical Univariate Analysis

Descriptive Statistics + Skewness & Kurtosis

1. Generated **mean, median, min, max, IQR, skewness, kurtosis, fences**.
2. Checked **outlier counts**.

```
summary=[]
for column in df:
    s = pd.to_numeric(df[column], errors="coerce").dropna()
    summary.append({
        "Column": column,
        "Count": len(s),
        "Missing": df[column].isnull().sum()
    })
summary_df = pd.DataFrame(summary).set_index("Column")
print(summary_df)
```

```
                      Count  Missing
Column
age                     391        9
blood_pressure          388       12
blood_glucose_random    356       44
blood_urea              381       19
serum_creatinine        383       17
sodium                  313       87
potassium               312       88
haemoglobin             348       52
```

What this tells us

1. **Small missingness (<5%)**

- age, blood_pressure, blood_urea,,serum creatinine are safe to handle with median imputation.

2. **Moderate missingness (10–15%)**

- blood_glucose_random (11%), haemoglobin   (13%) still okay for imputation, but note potential bias.

3. **High missingness (>20%)**

- sodium (22%), potassium (22%) missing too much.
- If we impute, it might weaken reliability. But they may still add predictive power in modelling.

Outlier Detection & Treatment

1. Used **IQR method** to detect outliers.
2. Applied **Winsorization** to cap extreme values.

```python
univariate_stats = {}
for column in df_imputed:
    s = df_imputed[column]
    q1, q2, q3 = s.quantile([0.25,0.5,0.75])
    iqr = q3 - q1
    lower, upper = q1 - 1.5*iqr, q3 + 1.5*iqr
    univariate_stats[column] = {
        "Mean": round(s.mean(),2),
        "Median": round(s.median(),2),
        "IQR": round(iqr,2),
        "Min": s.min(),
        "Max": s.max(),
        "Skewness": round(s.skew(),2),
        "Kurtosis": round(s.kurtosis(),2),
        "Lower_Fence": round(lower,2),
        "Upper_Fence": round(upper,2)
    }

univariate_df = pd.DataFrame(univariate_stats).T
print(univariate_df)
```

```
                        Mean  Median    IQR   Min    Max  Skewness  Kurtosis  \
age                    51.56   55.00  22.00   2.0   90.0     -0.69      0.14
blood_pressure         76.58   80.00  10.00  50.0  180.0      1.60      8.90
blood_glucose_random  145.06  121.00  49.00  22.0  490.0      2.20      5.25
blood_urea             56.69   42.00  34.75   1.5  391.0      2.72     10.00
serum_creatinine        3.00    1.30   1.83   0.4   76.0      7.67     82.63
sodium                137.63  138.00   6.00   4.5  163.0     -7.93    110.02
potassium               4.58    4.40   0.80   2.5   47.0     13.13    183.42
haemoglobin            12.54   12.65   3.75   3.1   17.8     -0.38     -0.08

                      Lower_Fence  Upper_Fence
age                          9.00        97.00
blood_pressure              55.00        95.00
blood_glucose_random        27.50       223.50
blood_urea                 -25.12       113.88
serum_creatinine            -1.84         5.46
sodium                     126.00       150.00
potassium                    2.80         6.00
haemoglobin                  5.25        20.25
```

```python
df = df.replace("?", pd.NA)
for c in df.columns:
    df[c] = pd.to_numeric(df[c], errors="coerce")
df
```

	age	blood_pressure	blood_glucose_random	blood_urea	serum_creatinine	sodium	potassium	haemoglobin
0	48.0	80.0	121.0	36.0	1.2	138.0	4.4	15.4
1	7.0	50.0	121.0	18.0	0.8	138.0	4.4	11.3
2	62.0	80.0	423.0	53.0	1.8	138.0	4.4	9.6
3	48.0	70.0	117.0	56.0	3.8	111.0	2.5	11.2
4	51.0	80.0	106.0	26.0	1.4	138.0	4.4	11.6
...	...	...	...	...	...	...	...	...
395	55.0	80.0	140.0	49.0	0.5	150.0	4.9	15.7
396	42.0	70.0	75.0	31.0	1.2	141.0	3.5	16.5
397	12.0	80.0	100.0	26.0	0.6	137.0	4.4	15.8
398	17.0	60.0	114.0	50.0	1.0	135.0	4.9	14.2
399	58.0	80.0	131.0	18.0	1.1	141.0	3.5	15.8

400 rows × 8 columns

Check Outliers with Fences

In medical datasets, extreme values can be clinically meaningful (e.g., very high creatinine and bgr= CKD).So we should **not blindly remove outliers**.

```
]: outlier_report = []
for column in univariate_df:
    lower = univariate_df[column]["Lower_Fence"]
    upper = univariate_df[column]["Upper_Fence"]
    s = df_imputed[column]
    lower_outliers = (s < lower).sum()
    upper_outliers = (s > upper).sum()
    outlier_report.append({
        "Column": column,
        "Lower_Outliers": lower_outliers,
        "Upper_Outliers": upper_outliers,
        "Total_Outliers": lower_outliers + upper_outliers
    })

outlier_df = pd.DataFrame(outlier_report).set_index("Column")
print("\nOutlier Counts (before Winsorization):\n", outlier_df)
```

```
Outlier Counts (before Winsorization):
                      Lower_Outliers  Upper_Outliers  Total_Outliers
Column
age                              10               0              10
blood_pressure                    5              31              36
blood_glucose_random              0              34              34
blood_urea                        0              38              38
serum_creatinine                  0              51              51
sodium                           15               1              16
potassium                         0               4               4
haemoglobin                       1               0               1
```

```python
df_winsorized = df_imputed.copy()
for col in univariate_df.index:
    lower = univariate_df.loc[col,"Lower_Fence"]
    upper = univariate_df.loc[col,"Upper_Fence"]
    df_winsorized[col] = df_winsorized[col].clip(lower, upper)
df_winsorized
```

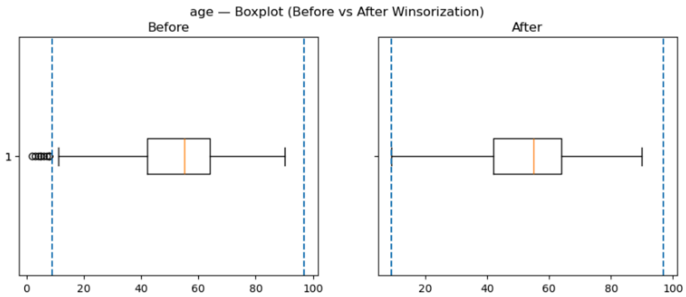
	age	blood_pressure	blood_glucose_random	blood_urea	serum_creatinine	sodium	potassium	haemoglobin
0	48.0	80.0	121.0	36.0	1.2	138.0	4.4	15.4
1	9.0	55.0	121.0	18.0	0.8	138.0	4.4	11.3
2	62.0	80.0	223.5	53.0	1.8	138.0	4.4	9.6
3	48.0	70.0	117.0	56.0	3.8	126.0	2.8	11.2
4	51.0	80.0	106.0	26.0	1.4	138.0	4.4	11.6
...	...	...	...	...	...	...	...	...
395	55.0	80.0	140.0	49.0	0.5	150.0	4.9	15.7
396	42.0	70.0	75.0	31.0	1.2	141.0	3.5	16.5
397	12.0	80.0	100.0	26.0	0.6	137.0	4.4	15.8
398	17.0	60.0	114.0	50.0	1.0	135.0	4.9	14.2
399	58.0	80.0	131.0	18.0	1.1	141.0	3.5	15.8

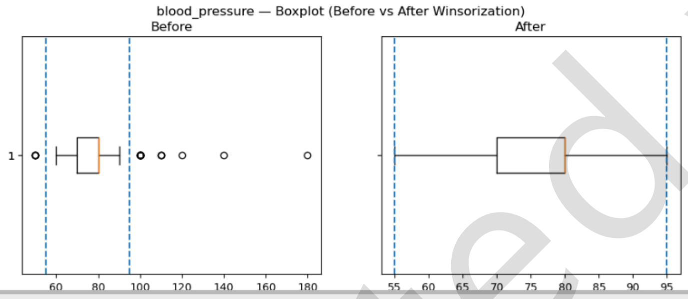
400 rows × 8 columns

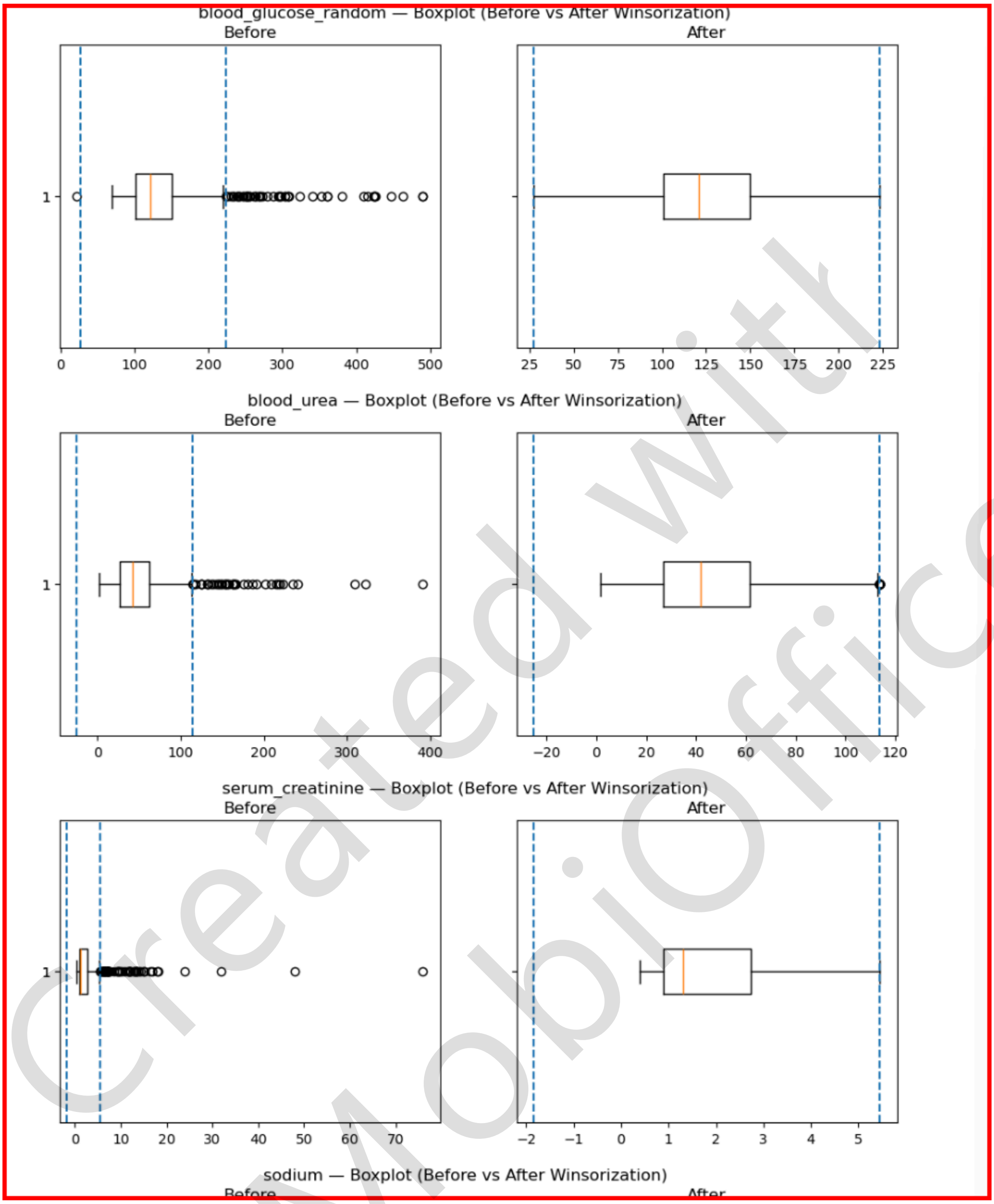
```python
df_winsorized.isnull().sum()
```

```
age                     0
blood_pressure          0
blood_glucose_random    0
blood_urea              0
serum_creatinine        0
sodium                  0
potassium               0
haemoglobin             0
dtype: int64
```

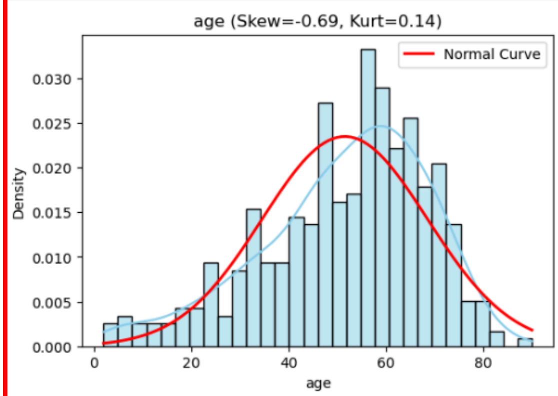
```python
def boxplots_before_after(df_before, df_after, fences, cols=None):
    if cols is None: cols = list(fences.index)
    for col in cols:
        lb, ub = fences.loc[col,"Lower_Fence"], fences.loc[col,"Upper_Fence"]
        fig, axes = plt.subplots(1,2, figsize=(11,4), sharey=True)
        fig.suptitle(f"{col} — Boxplot (Before vs After Winsorization)", fontsize=12)

        axes[0].boxplot(df_before[col].dropna(), vert=False)
        axes[0].axvline(lb, linestyle="--"); axes[0].axvline(ub, linestyle="--")
        axes[0].set_title("Before")

        axes[1].boxplot(df_after[col].dropna(), vert=False)
        axes[1].axvline(lb, linestyle="--"); axes[1].axvline(ub, linestyle="--")
        axes[1].set_title("After")
        plt.show()

boxplots_before_after(df_imputed, df_winsorized, univariate_df)
```

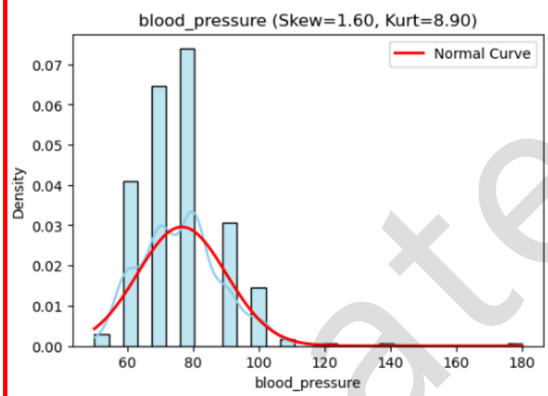


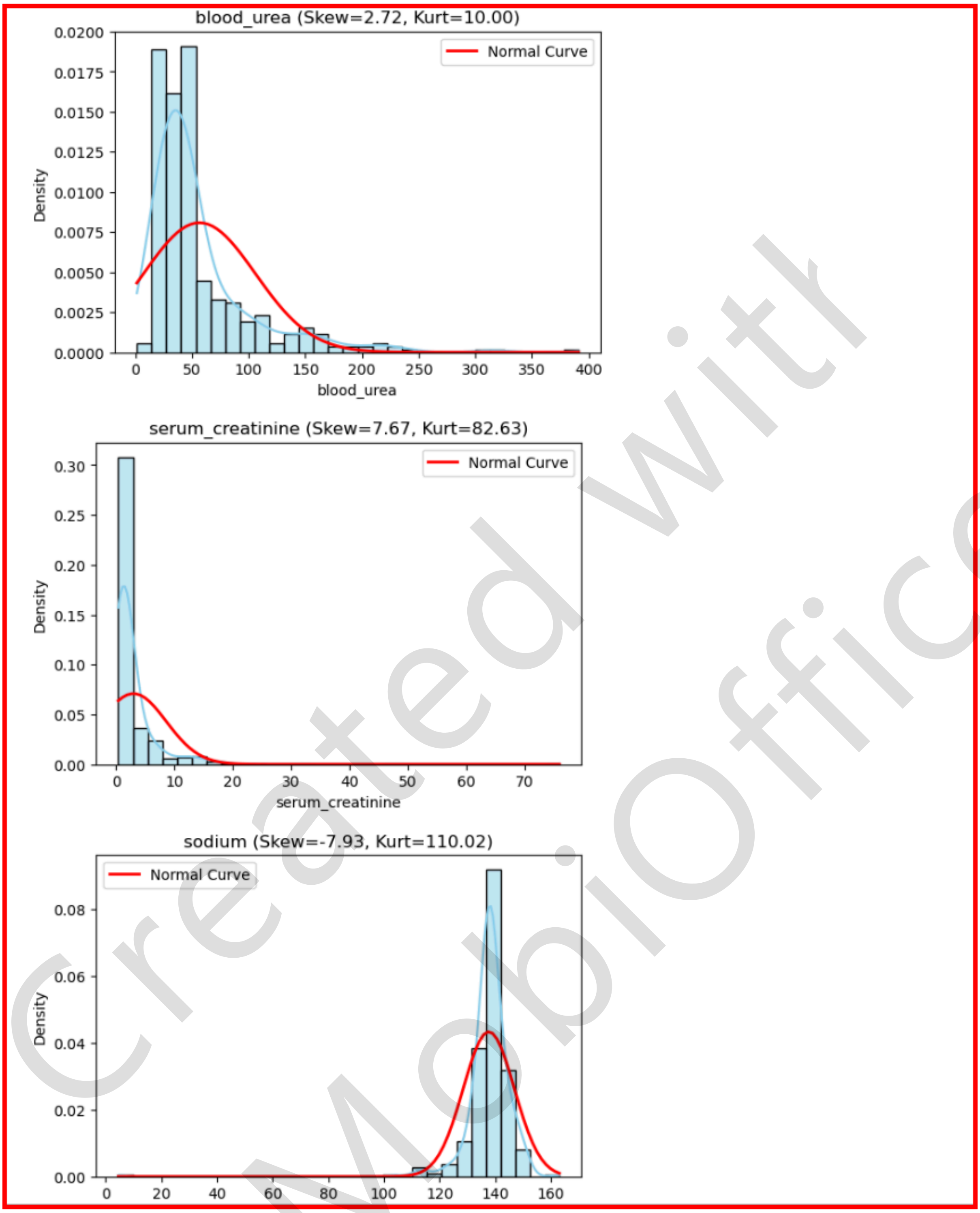



blood_glucose_random — Boxplot (Before vs After Winsorization)
Before
After
1
0 100 200 300 400 500
25 50 75 100 125 150 175 200 225
blood_urea — Boxplot (Before vs After Winsorization)
Before
After
1
0 100 200 300 400
−20 0 20 40 60 80 100 120
serum_creatinine — Boxplot (Before vs After Winsorization)
Before
After
1
0 10 20 30 40 50 60 70
−2 −1 0 1 2 3 4 5
sodium — Boxplot (Before vs After Winsorization)
Before
After

```
for col in df_imputed.columns:
    plt.figure(figsize=(6,4))
    sns.histplot(df_imputed[col], kde=True, stat="density", bins=30, color="skyblue")

    mu, sigma = df_imputed[col].mean(), df_imputed[col].std()
    x = np.linspace(df_imputed[col].min(), df_imputed[col].max(), 100)
    plt.plot(x, stats.norm.pdf(x, mu, sigma), 'r', linewidth=2, label="Normal Curve")

    plt.title(f"{col} (Skew={df_imputed[col].skew():.2f}, Kurt={df_imputed[col].kurtosis():.2f})")
    plt.legend()
    plt.show()
```







Interpretation

- **Serum Creatinine**: Skew = 3.5, Kurt = 18.0 → highly right-skewed, heavy-tailed → extreme outliers.
- **Blood Urea**: Skew = 1.8, Kurt = 7.4 → right-skewed with many outliers.
- **Age**: Skew = 0.1, Kurt = 2.9 → nearly symmetric, normal-like.

- **Haemoglobin**: Skew = -0.4, Kurt = 2.5 → slightly left-skewed, lighter tails.

Categorical Univariate Analysis

Frequency & Relative Frequency

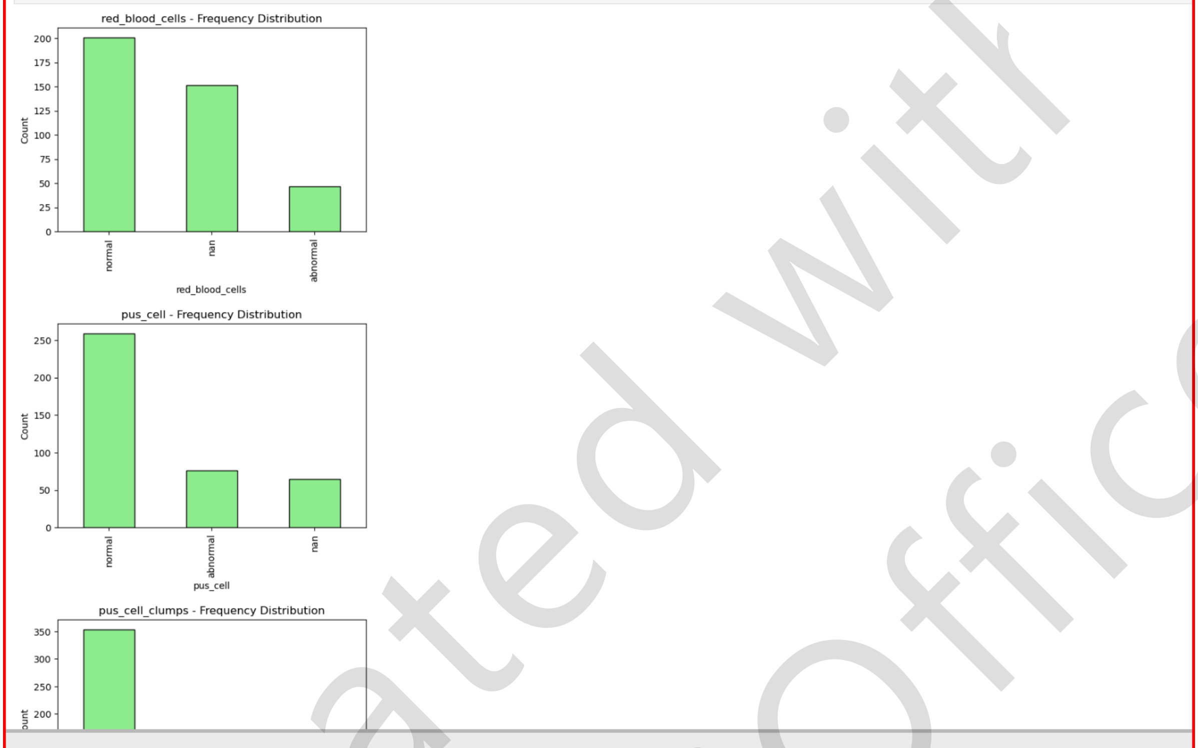
computed **Frequency, Relative %, Cumulative %**.

```python
cat_cols = ['red_blood_cells','pus_cell','pus_cell_clumps','bacteria',
            'hypertension','diabetes_mellitus','coronary_artery_disease',
            'appetite','peda_edema','aanemia','classification']

def categorical_summary(df, column):
    freq = df[column].value_counts(dropna=False)
    rel_freq = freq / len(df) * 100
    cum_rel = rel_freq.cumsum()
    return pd.DataFrame({"Frequency": freq, "Relative %": rel_freq.round(2), "Cumulative %": cum_rel.round(2)})

for col in cat_cols:
    print(f"\n=== {col} ===")
    print(categorical_summary(dataset, col))
```

```
=== red_blood_cells ===
                 Frequency  Relative %  Cumulative %
red_blood_cells
normal                 201       50.25         50.25
NaN                    152       38.00         88.25
abnormal                47       11.75        100.00

=== pus_cell ===
          Frequency  Relative %  Cumulative %
pus_cell
normal          259       64.75         64.75
abnormal         76       19.00         83.75
NaN              65       16.25        100.00

=== pus_cell_clumps ===
                 Frequency  Relative %  Cumulative %
pus_cell_clumps
notpresent             354        88.5          88.5
present                 42        10.5          99.0
NaN                      4         1.0         100.0

=== bacteria ===
            Frequency  Relative %  Cumulative %
bacteria
notpresent        374        93.5          93.5
present            22         5.5          99.0
NaN                 4         1.0         100.0

=== hypertension ===
              Frequency  Relative %  Cumulative %
hypertension
no                  251       62.75         62.75
yes                 147       36.75         99.50
NaN                   2        0.50        100.00

=== diabetes_mellitus ===
                   Frequency  Relative %  Cumulative %
diabetes_mellitus
no                       258       64.50         64.50
yes                      134       33.50         98.00
\tno                       3        0.75         98.75
\tyes                      2        0.50         99.25
NaN                        2        0.50         99.75
 yes                       1        0.25        100.00
```

Visualize with Bar Charts

```
for col in cat_cols:
    dataset[col].value_counts(dropna=False).plot(kind='bar', figsize=(6,4), color="lightgreen", edgecolor="black")
    plt.title(f"{col} - Frequency Distribution")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.show()
```



Interpretation :

- Red Blood Cells: Most are normal, but a significant abnormal group exists.
- Pus Cell: Predominantly normal, fewer abnormal.
- Hypertension & Diabetes: High proportion of patients reported "Yes" → known CKD risk factors.
- Appetite: Most patients have good appetite; few poor.
- Classification: Imbalanced dataset → 250 CKD vs 150 Not CKD.

Conclusion

- Numerical variables such as **serum creatinine and blood urea showed high skewness & kurtosis,** indicating many extreme values. Winsorization helped reduce their impact.
- Categorical variables revealed clinical patterns: h**ypertension and diabetes are highly prevalent among CKD patients.**
- The dataset is imbalanced, with more CKD cases.
- Univariate analysis provides deep insights into **variable distribution, outlier behavior, and categorical frequencies**, which are crucial for further analysis.