

Kidney Disease Dataset

Data Preprocessing and Visualization

Student: Hassan Saud Al-Soleiss

This notebook handles missing values, treats outliers using the IQR method, and creates data visualizations.

Import libraries and load dataset

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("kidney_disease.csv")

print("Dataset shape:", data.shape)
data.head()
```

Dataset shape: (398, 25)

```
Out[1]:
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	w
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	780
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	600
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	750
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	670
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	730

5 rows × 25 columns



Clean text values and convert numeric columns

```
In [2]: text_columns = data.select_dtypes(include="object").columns

for column in text_columns:
    data[column] = data[column].astype(str).str.strip()
    data[column] = data[column].replace(["?", "nan", ""], np.nan)

numeric_text_columns = ["pcv", "wc", "rc"]

for column in numeric_text_columns:
```

```
data[column] = pd.to_numeric(data[column], errors="coerce")
```

```
data.dtypes
```

```
Out[2]: 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         float64
wc          float64
rc          float64
htn         object
dm          object
cad         object
appet       object
pe          object
ane         object
classification object
dtype: object
```

Check and handle missing values

```
In [5]: print("Missing values before treatment:")
print(data.isnull().sum())

numerical_columns = data.select_dtypes(include=np.number).columns
categorical_columns = data.select_dtypes(include="object").columns

for column in numerical_columns:
    data[column] = data[column].fillna(data[column].median())

for column in categorical_columns:
    data[column] = data[column].fillna(data[column].mode()[0])

print("\nTotal missing values after treatment:")
print(data.isnull().sum().sum())
```

Missing values before treatment:

```
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
```

Total missing values after treatment:

0

Handle outliers using IQR capping

```
In [6]: for column in numerical_columns:
        q1 = data[column].quantile(0.25)
        q3 = data[column].quantile(0.75)
        iqr = q3 - q1

        lower_limit = q1 - 1.5 * iqr
        upper_limit = q3 + 1.5 * iqr

        data[column] = data[column].clip(
            lower=lower_limit,
            upper=upper_limit
        )

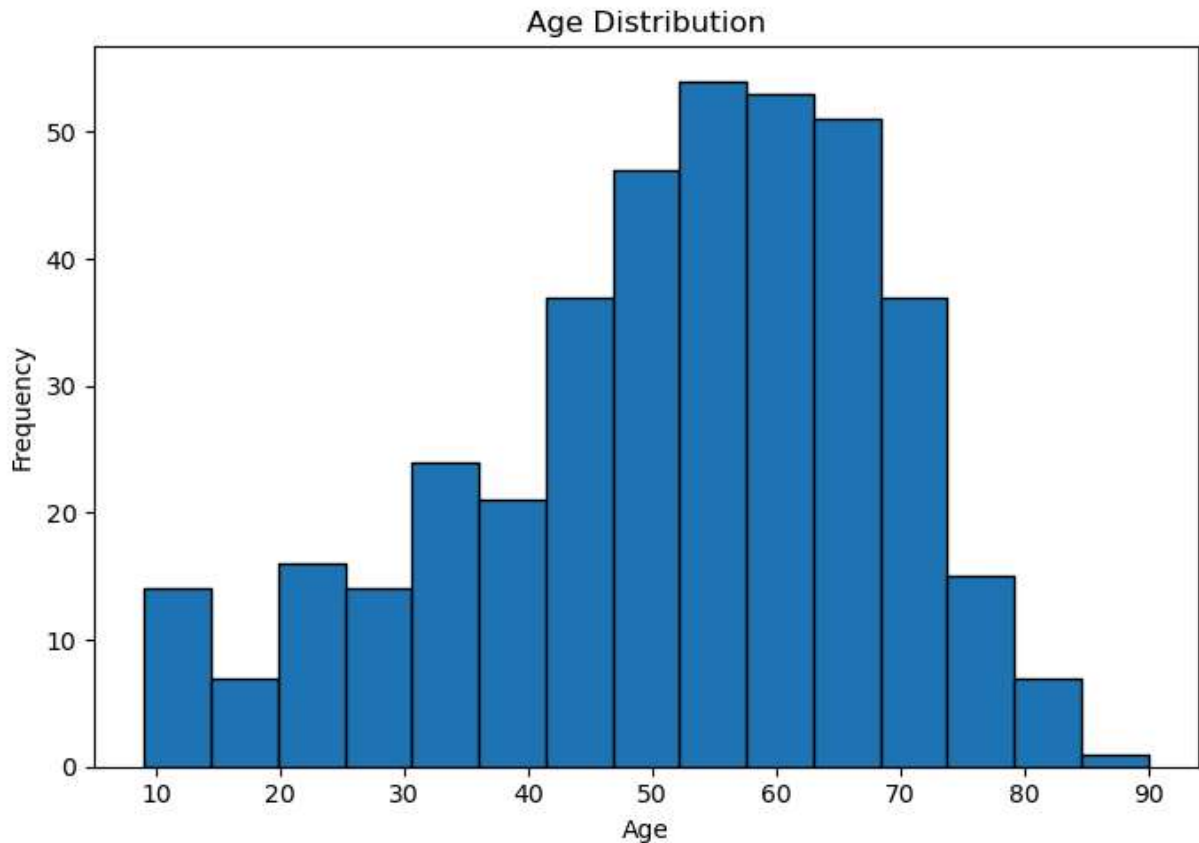
        print("Outliers were handled using IQR capping.")
```

Outliers were handled using IQR capping.

Age distribution

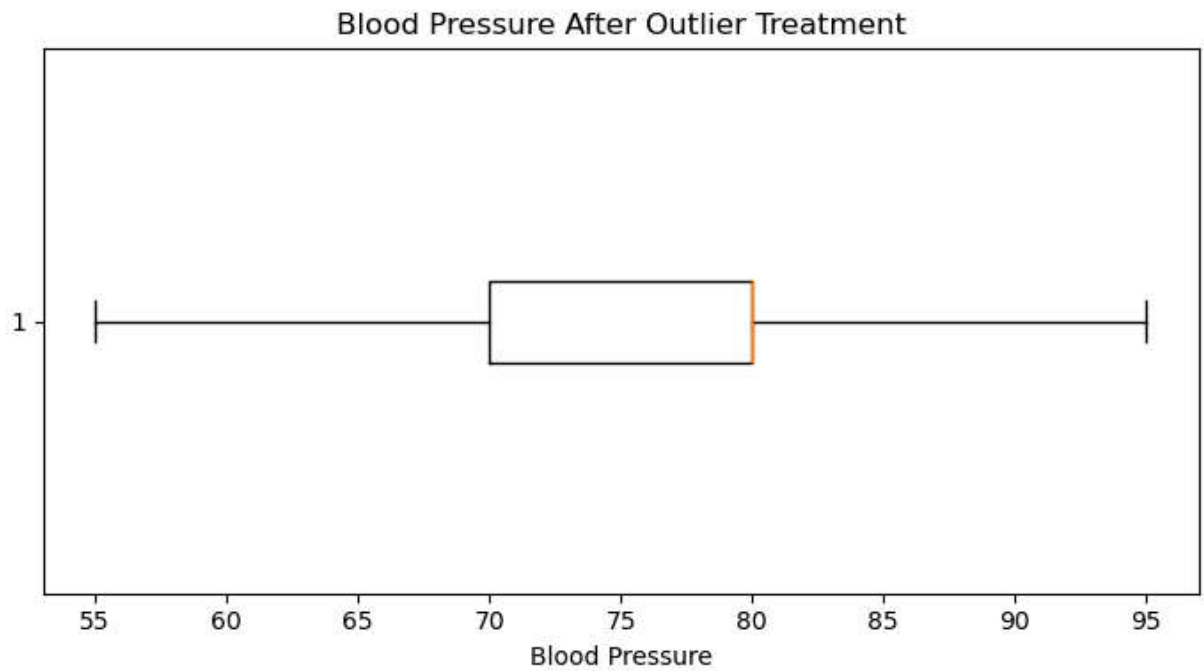
```
In [7]: plt.figure(figsize=(7, 5))
        plt.hist(data["age"], bins=15, edgecolor="black")
```

```
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



Blood pressure boxplot

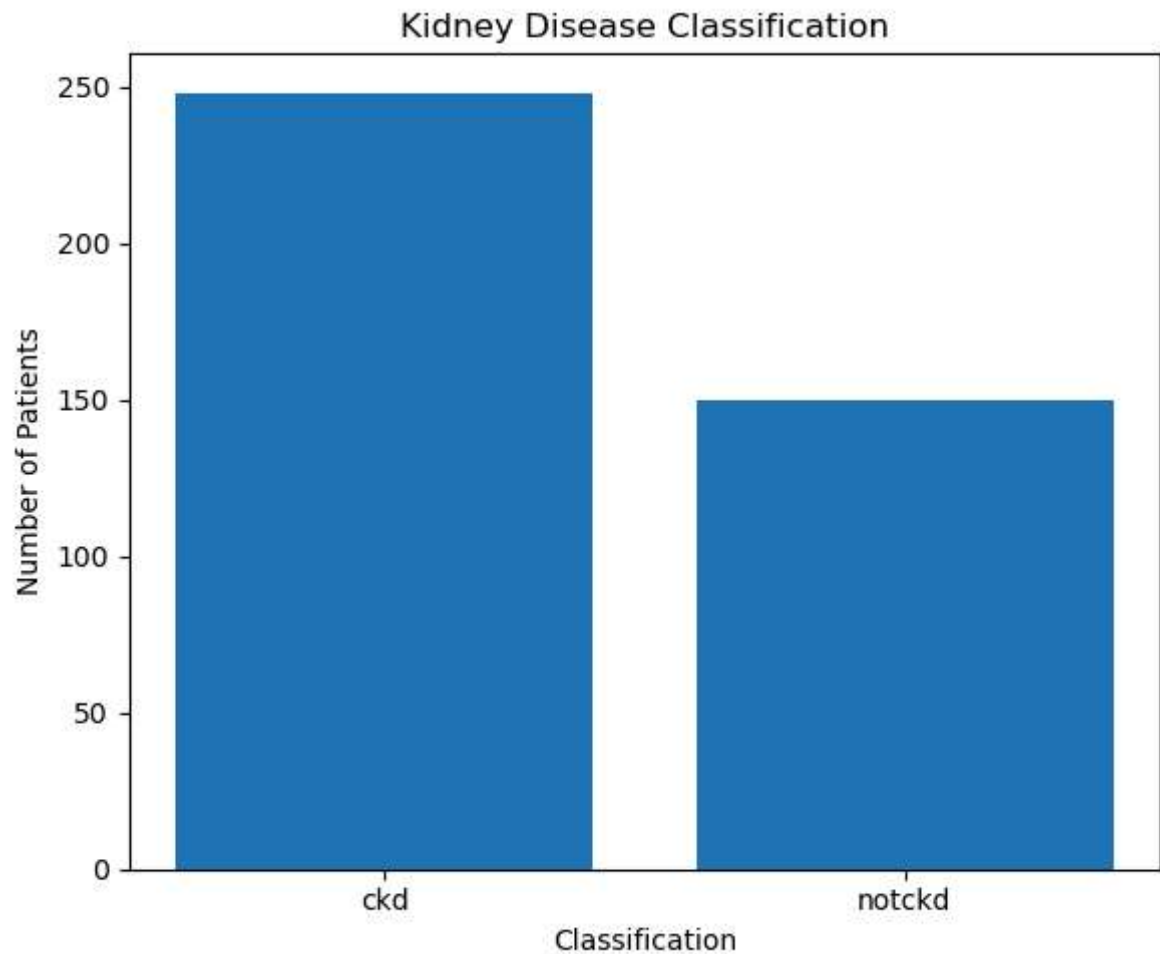
```
In [8]: plt.figure(figsize=(7, 4))
plt.boxplot(data["bp"], vert=False)
plt.title("Blood Pressure After Outlier Treatment")
plt.xlabel("Blood Pressure")
plt.tight_layout()
plt.show()
```



Classification count

```
In [9]: class_counts = data["classification"].value_counts()

plt.figure(figsize=(6, 5))
plt.bar(class_counts.index, class_counts.values)
plt.title("Kidney Disease Classification")
plt.xlabel("Classification")
plt.ylabel("Number of Patients")
plt.tight_layout()
plt.show()
```



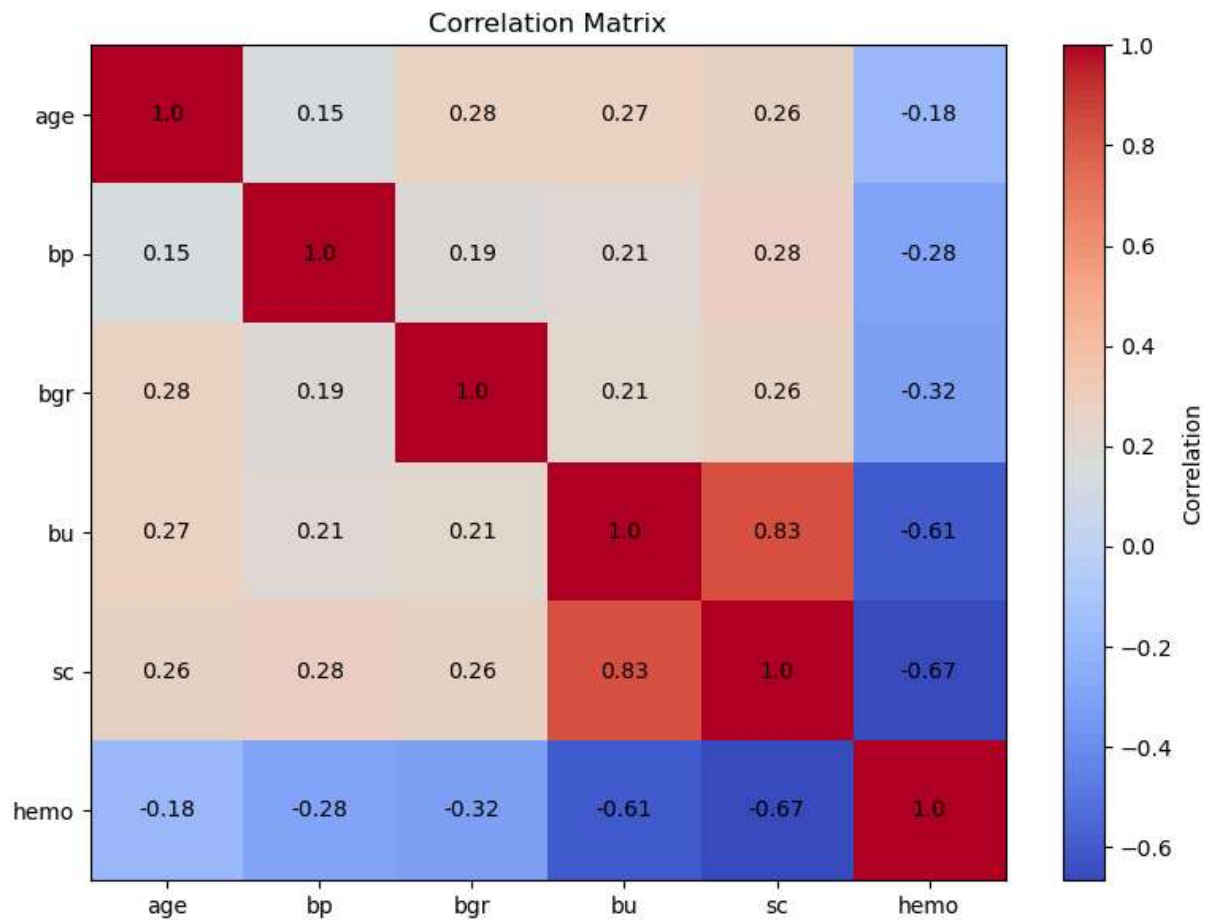
Correlation matrix

```
In [10]: selected_columns = ["age", "bp", "bgr", "bu", "sc", "hemo"]
correlation = data[selected_columns].corr()

plt.figure(figsize=(8, 6))
plt.imshow(correlation, cmap="coolwarm", aspect="auto")
plt.colorbar(label="Correlation")
plt.xticks(range(len(selected_columns)), selected_columns)
plt.yticks(range(len(selected_columns)), selected_columns)

for i in range(len(selected_columns)):
    for j in range(len(selected_columns)):
        plt.text(j, i, round(correlation.iloc[i, j], 2),
                 ha="center", va="center")

plt.title("Correlation Matrix")
plt.tight_layout()
plt.show()
```



Save cleaned dataset

```
In [11]: data.to_csv("kidney_disease_cleaned.csv", index=False)
          print("Cleaned dataset saved successfully.")
```

Cleaned dataset saved successfully.

```
In [ ]:
```