

```

from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('/content/drive/MyDrive/Projects/Project1/kidney_disease.csv')

### Handle missing values.

df.isnull().sum()
missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")

### Handle outliers.

data=df.copy()

# Automatically detect numerical columns
numeric_cols = data.select_dtypes(include=['number']).columns

def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Return a boolean series indicating if the value is an outlier
    return (column < lower_bound) | (column > upper_bound)
# Apply the IQR method to each feature in the DataFrame (Only for numerical columns)
for col in numeric_cols:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])
# Filter the DataFrame to display rows where any outlier flag is True

```

```
outliers_only = data[data.filter(like='_Outlier').any(axis=1)] # Display the filtered DataFrame
with outliers
outliers_only
```

```
data=df.copy()
def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)
# Replace outliers with NaN for each feature
for col in numeric_cols:
    data[col] = replace_outliers_with_nan(data[col])
null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

```
from sklearn.impute import KNNImputer
```

```
imputer = KNNImputer(n_neighbors=2)
```

```
# Impute only numerical columns
data[numeric_cols] = imputer.fit_transform(data[numeric_cols])
```

```
print(data[numeric_cols].isnull().sum())
```

```
### Perform data visualization.
```

```
#Correlation Heatmap
```

```
plt.figure(figsize=(12,10))
```

```
corr = data.select_dtypes(include='number').corr()
```

```
sns.heatmap(
    corr,
    cmap='viridis'
)
```

```
plt.title("Feature Correlation Heatmap")
```

```
plt.show()
```

```
# Bar Chart
```

```
plt.figure(figsize=(8,6))
```

```
sns.barplot(  
    x='classification',  
    y='bgr',  
    data=data  
)
```

```
plt.title("Average Blood Glucose by Classification")
```

```
plt.xlabel("Classification")
```

```
plt.ylabel("Average Blood Glucose")
```

```
plt.show()
```

```
# Histogram
```

```
plt.figure(figsize=(8,6))
```

```
sns.histplot(  
    data['bu'],  
    bins=20,  
    kde=True  
)
```

```
plt.title("Blood Urea Distribution")
```

```
plt.xlabel("Blood Urea")
```

```
plt.ylabel("Frequency")
```

```
plt.show()
```

```
# Pie Chart
```

```
plt.figure(figsize=(6,6))
```

```
data['htn'].value_counts().plot(  
    kind='pie',  
    autopct='%1.1f%%',  
    startangle=90  
)
```

```
plt.title("Hypertension Distribution")
```

```
plt.ylabel("")
```

```
plt.show()
```

```
# Line Plot
```

```
plt.figure(figsize=(8,6))
```

```
sns.lineplot(  
    x='age',  
    y='sc',  
    data=data  
)
```

```
plt.title("Serum Creatinine vs Age")
```

```
plt.xlabel("Age")
```

```
plt.ylabel("Serum Creatinine")
```

```
plt.show()
```

```
#Count Plot
```

```
plt.figure(figsize=(8,6))
```

```
sns.countplot(  
    x='dm',  
    data=data  
)
```

```
plt.title("Diabetes Mellitus Distribution")
```

```
plt.xlabel("Diabetes Mellitus")
```

```
plt.ylabel("Count")
```

```
plt.show()
```

```
#Scatter Plot
```

```
plt.figure(figsize=(8,6))
```

```
sns.scatterplot(  
    x='bu',  
    y='sc',  
    hue='classification',  
    data=data  
)
```

```
plt.title("Blood Urea vs Serum Creatinine")
plt.xlabel("Blood Urea")
plt.ylabel("Serum Creatinine")

plt.show()
```

```
# Box Plot
```

```
plt.figure(figsize=(8,6))
```

```
sns.boxplot(
    x='htn',
    y='hemo',
    data=data
)
```

```
plt.title("Hemoglobin by Hypertension Status")
plt.xlabel("Hypertension")
plt.ylabel("Hemoglobin")

plt.show()
```

```
df = pd.read_csv('/content/drive/MyDrive/Projects/Project1/heart.csv')
```

```
### Handle missing values.
```

```
df.isnull().sum()
missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

```
### Handle outliers.
```

```
data = df.copy()
```

```
# Automatically detect numerical columns
numeric_cols = data.select_dtypes(include=['number']).columns
```

```

def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    return (column < lower_bound) | (column > upper_bound)

# Display rows containing outliers
for col in numeric_cols:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])

outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
print(outliers_only)

# Remove outliers
data = df.copy()

for col in numeric_cols:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    data = data[
        (data[col] >= lower_bound) &
        (data[col] <= upper_bound)
    ]

print("Original dataset shape:", df.shape)
print("Dataset shape after removing outliers:", data.shape)

# Verify no remaining outliers
print("\nRemaining outliers:")
for col in numeric_cols:
    print(f'{col}: {detect_outliers_iqr(data[col]).sum()}')

### Perform data visualization.

# Correlation Heatmap

```

```
plt.figure(figsize=(12,10))

corr = data.select_dtypes(include='number').corr()

sns.heatmap(
    corr,
    cmap='viridis'
)

plt.title("Feature Correlation Heatmap")

plt.show()


# Bar Chart    Average Cholesterol by Heart Disease

plt.figure(figsize=(8,6))

sns.barplot(
    x='target',
    y='chol',
    data=data
)

plt.title("Average Cholesterol by Heart Disease")
plt.xlabel("Heart Disease (Target)")
plt.ylabel("Average Cholesterol")

plt.show()


# Histogram    Maximum Heart Rate Distribution

plt.figure(figsize=(8,6))

sns.histplot(
    data['thalach'],
    bins=20,
    kde=True
)

plt.title("Maximum Heart Rate Distribution")
plt.xlabel("Maximum Heart Rate")
plt.ylabel("Frequency")

plt.show()
```

```
#Pie Chart    Gender Distribution
```

```
plt.figure(figsize=(6,6))

data['sex'].value_counts().plot(
    kind='pie',
    autopct='%1.1f%%',
    startangle=90
)

plt.title("Gender Distribution")
plt.ylabel("")

plt.show()
```

```
# Line Plot    Cholesterol vs Age
```

```
plt.figure(figsize=(8,6))

sns.lineplot(
    x='age',
    y='chol',
    data=data
)

plt.title("Cholesterol vs Age")
plt.xlabel("Age")
plt.ylabel("Cholesterol")

plt.show()
```

```
# Count Plot    Heart Disease Counts
```

```
plt.figure(figsize=(8,6))

sns.countplot(
    x='target',
    data=data
)

plt.title("Heart Disease Distribution")
plt.xlabel("Heart Disease (Target)")
plt.ylabel("Count")
```

```
plt.show()
```

```
# Scatter Plot    Cholesterol vs Maximum Heart Rate
```

```
plt.figure(figsize=(8,6))
```

```
sns.scatterplot(  
    x='chol',  
    y='thalach',  
    hue='target',  
    data=data  
)
```

```
plt.title("Cholesterol vs Maximum Heart Rate")
```

```
plt.xlabel("Cholesterol")
```

```
plt.ylabel("Maximum Heart Rate")
```

```
plt.show()
```

```
# Box Plot        Resting Blood Pressure by Heart Disease
```

```
plt.figure(figsize=(8,6))
```

```
sns.boxplot(  
    x='target',  
    y='trestbps',  
    data=data  
)
```

```
plt.title("Resting Blood Pressure by Heart Disease")
```

```
plt.xlabel("Heart Disease (Target)")
```

```
plt.ylabel("Resting Blood Pressure")
```

```
plt.show()
```