

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import KNNImputer
import warnings
(warnings.filterwarnings('ignore

("sns.set_style("whitegrid
(plt.rcParams['figure.figsize'] = (12, 6

(print("=" * 80
(print("PROJECT 1: DATA CLEANING AND VISUALIZATION
(print("=" * 80

(...print("\n[STEP 1] Loading Kidney Disease Dataset
(df = pd.read_csv('kidney_disease.csv

(!print(f"\n✓ Dataset loaded successfully
(print(f" - Shape: {df.shape[0]} rows, {df.shape[1]} columns
(:print(f"\nFirst few rows of the dataset
(())print(df.head

(:print(f"\n\nDataset Information
("{print(f"Data Types:\n{df.dtypes

(print("\n" + "=" * 80
(print("PART 2: IDENTIFYING MISSING VALUES
(print("=" * 80

Calculate missing values #
()missing_data = df.isnull().sum
missing_percentage = (df.isnull().sum() / len(df)) * 100

})missing_df = pd.DataFrame
,Column': missing_data.index'
,Missing Count': missing_data.values'
Percentage': missing_percentage.values'
({

missing_df = missing_df[missing_df['Missing Count'] > 0].sort_values('Percentage',
(ascending=False

(:print("\nMissing Values Summary
((print(missing_df.to_string(index=False

```

```

(print("\n" + "=" * 80)
(print("PART 3: HANDLING MISSING VALUES
(print("=" * 80

(...print("\n[STEP 3.1] Removing columns with >30% missing data

threshold = 30
(columns_to_drop = missing_percentage[missing_percentage > threshold].index.tolist

:if len(columns_to_drop) > 0
("{print(f" Columns to remove: {columns_to_drop
(print(f" Reason: They have >30% missing data
(df = df.drop(columns=columns_to_drop
(print(f"✓ Removed {len(columns_to_drop)} columns
:else
("(print(f"✓ No columns need to be removed (all have <30% missing

("{print(f"\nDataset shape after dropping: {df.shape

(...print("\n[STEP 3.2] Filling remaining missing values

(numerical_cols = df.select_dtypes(include=[np.number]).columns.tolist
(categorical_cols = df.select_dtypes(include=['object']).columns.tolist

("{print(f"\nNumerical columns: {numerical_cols
("{print(f"Categorical columns: {categorical_cols

:for col in numerical_cols
:if df[col].isnull().sum() > 0
(mean_val = df[col].mean
(df[col].fillna(mean_val, inplace=True
("{print(f" ✓ Filled {col} with mean value: {mean_val:.2f

:for col in categorical_cols
:if df[col].isnull().sum() > 0
(mode_val = df[col].mode()[0
(df[col].fillna(mode_val, inplace=True
("{print(f" ✓ Filled {col} with mode value: {mode_val

(":print(f"\n✓ Missing values check after handling
("{print(f" Total missing values remaining: {df.isnull().sum().sum

(print("\n" + "=" * 80
(print("PART 4: DETECTING AND HANDLING OUTLIERS (IQR Method
(print("=" * 80

```

```

"?print("\nWhy IQR Method
(print(" - Works well with medical data (often skewed
(print(" - Robust to extreme values
(print(" - Easier to interpret and explain
(print(" - Formula: Lower Bound =  $Q1 - 1.5 \times IQR$ 
(print("      Upper Bound =  $Q3 + 1.5 \times IQR$ 

:(def detect_outliers_iqr(data_series
(Q1 = data_series.quantile(0.25
(Q3 = data_series.quantile(0.75
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

(outliers = (data_series < lower_bound) | (data_series > upper_bound
return outliers, lower_bound, upper_bound

:(def replace_outliers_with_nan(df_input, column
([outliers, lower, upper = detect_outliers_iqr(df_input[column
:if outliers.sum() > 0
df_input.loc[outliers, column] = np.nan
(":{print(f"\n{column
("{print(f" Q1: {df_input[column].quantile(0.25):.2f
("{print(f" Q3: {df_input[column].quantile(0.75):.2f
("{print(f" Lower Bound: {lower:.2f
("{print(f" Upper Bound: {upper:.2f
("{()print(f" Outliers detected: {outliers.sum

(print("\n[STEP 4.1] Detecting outliers in numerical columns...\n

(df_copy = df.copy

:for col in numerical_cols
(replace_outliers_with_nan(df, col

(print("\n[STEP 4.2] Filling outlier values with median...\n

:for col in numerical_cols
:if df[col].isnull().sum() > 0
(median_val = df[col].median
(df[col].fillna(median_val, inplace=True
("{print(f" ✓ {col}: Filled {df[col].isnull().sum()} values with median {median_val:.2f

("!print("\n✓ All outliers have been handled

(print("\n" + "=" * 80

```

```

("print("PART 5: DATA VISUALIZATION
(print("=" * 80

"...print("\nCreating visualizations to understand the data

(fig, axes = plt.subplots(1, 2, figsize=(14, 5

(missing_data_before = pd.read_csv('kidney_disease.csv').isnull().sum
missing_data_before = missing_data_before[missing_data_before >
(0)].sort_values(ascending=True

:if len(missing_data_before) > 0
('axes[0].barh(range(len(missing_data_before)), missing_data_before.values, color='coral
(((axes[0].set_yticks(range(len(missing_data_before
(axes[0].set_yticklabels(missing_data_before.index
('axes[0].set_xlabel('Number of Missing Values
('axes[0].set_title('Missing Values BEFORE Cleaning', fontsize=12, fontweight='bold
(axes[0].grid(axis='x', alpha=0.3

,'axes[1].text(0.5, 0.5, '✓ ALL MISSING VALUES\nHAVE BEEN HANDLED
,'ha='center', va='center', fontsize=16, fontweight='bold
((bbox=dict(boxstyle='round', facecolor='lightgreen', alpha=0.7
('axes[1].set_title('Missing Values AFTER Cleaning', fontsize=12, fontweight='bold
('axes[1].axis('off

(plt.tight_layout
('plt.savefig('01_missing_values_comparison.png', dpi=300, bbox_inches='tight
("print("✓ Saved: 01_missing_values_comparison.png
(plt.close

"...print("\nCreating correlation heatmap
(plt.figure(figsize=(12, 10

(df_encoded = df.copy
:for col in categorical_cols
df_encoded[col] = pd.Categorical(df_encoded[col]).codes

(correlation_matrix = df_encoded.corr

,sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0
(fmt='.2f', square=True, linewidths=0.5
('plt.title('Correlation Heatmap - Kidney Disease Dataset', fontsize=14, fontweight='bold
(plt.tight_layout
('plt.savefig('02_correlation_heatmap.png', dpi=300, bbox_inches='tight
("print("✓ Saved: 02_correlation_heatmap.png
(plt.close

"...print("\nCreating distribution plots

```

```

(fig, axes = plt.subplots(2, 3, figsize=(15, 10
)axes = axes.ravel

[cols_to_plot = numerical_cols[:6

:(for idx, col in enumerate(cols_to_plot
(axes[idx].hist(df[col], bins=30, color='skyblue', edgecolor='black', alpha=0.7
('axes[idx].set_title(f'Distribution of {col}', fontweight='bold
(axes[idx].set_xlabel(col
('axes[idx].set_ylabel('Frequency
(axes[idx].grid(alpha=0.3

(mean_val = df[col].mean
axes[idx].axvline(mean_val, color='red', linestyle='--', linewidth=2, label=f'Mean:
('{{mean_val:.2f
(axes[idx].legend

(plt.tight_layout
(plt.savefig('03_distributions.png', dpi=300, bbox_inches='tight
("print("✓ Saved: 03_distributions.png
(plt.close

("...print("\nCreating outlier comparison plots
(fig, axes = plt.subplots(1, 2, figsize=(14, 6

(df_before = pd.read_csv('kidney_disease.csv').fillna(0
([df_before[numerical_cols].boxplot(ax=axes[0
('axes[0].set_title('Box Plots BEFORE Handling Outliers', fontsize=12, fontweight='bold
('axes[0].set_ylabel('Values
(axes[0].tick_params(axis='x', rotation=45

([df[numerical_cols].boxplot(ax=axes[1
('axes[1].set_title('Box Plots AFTER Handling Outliers', fontsize=12, fontweight='bold
('axes[1].set_ylabel('Values
(axes[1].tick_params(axis='x', rotation=45

(plt.tight_layout
(plt.savefig('04_outliers_comparison.png', dpi=300, bbox_inches='tight
("print("✓ Saved: 04_outliers_comparison.png
(plt.close

("...print("\nCreating classification analysis
:if 'classification' in df.columns
((fig, axes = plt.subplots(1, 2, figsize=(12, 5

Pie chart #
(counts = df['classification'].value_counts
, '%axes[0].pie(counts.values, labels=counts.index, autopct='%1.1f

```

```
(colors=['lightcoral', 'lightblue'], startangle=90
('axes[0].set_title('Classification Distribution (Pie Chart)', fontweight='bold
```

Bar chart #

```
(['counts.plot(kind='bar', ax=axes[1], color=['lightcoral', 'lightblue
('axes[1].set_title('Classification Distribution (Bar Chart)', fontweight='bold
('axes[1].set_xlabel('Classification
('axes[1].set_ylabel('Count
(axes[1].tick_params(axis='x', rotation=0
```

```
()plt.tight_layout
('plt.savefig('05_classification_distribution.png', dpi=300, bbox_inches='tight
("print("✓ Saved: 05_classification_distribution.png
()plt.close
```

```
(print("\n" + "=" * 80
("print("PROJECT COMPLETION SUMMARY
(print("=" * 80
```

```
"""print(f
```

```
=====rf
||=====rf
||                ✓ TASK 1: MISSING VALUES                ||
=====Lf
||=====
```

:Steps Completed

Identified all columns with missing values .1

:Removed columns with >30% missing data .2

{'Removed: {columns_to_drop if columns_to_drop else 'None -

:Filled remaining missing values .3

Numerical columns: Filled with MEAN value -

(Categorical columns: Filled with MODE (most frequent value -

:Result

{()}Missing values before: {pd.read_csv('kidney_disease.csv').isnull().sum().sum -

✓ {()}Missing values after: {df.isnull().sum().sum -

```
=====rf
||=====rf
||                ✓ TASK 2: OUTLIER HANDLING                ||
=====Lf
||=====
```

(Method Used: IQR (Interquartile Range

Reason: Medical data is often skewed, IQR is more robust -

:Formula -

Lower Bound = $Q1 - 1.5 \times IQR$ •

Upper Bound = $Q3 + 1.5 \times IQR$ •

Any value outside these bounds is an outlier •

:Process

(Calculated Q1 (25th percentile) and Q3 (75th percentile) .1

Identified outliers using IQR method .2

Replaced outliers with NaN .3

Filled NaN values with MEDIAN .4

:Result

Dataset cleaned and ready for analysis -

(Outliers handled gracefully (not deleted, replaced with median -

	✓ TASK 3: DATA VISUALIZATION

:Visualizations Created

missing_values_comparison.png_01 .1

Shows missing values before and after cleaning -

correlation_heatmap.png_02 .2

Displays correlation between all variables -

Red = positive correlation, Blue = negative correlation -

distributions.png_03 .3

Shows distribution of numerical features -

Helps identify data patterns -

outliers_comparison.png_04 .4

Box plots before and after outlier handling -

Visual proof of outlier treatment -

classification_distribution.png_05 .5

Shows target variable distribution -

	FINAL DATASET STATISTICS

Dataset Shape: {df.shape[0]} rows × {df.shape[1]} columns

```
:Numerical Columns Summary
{()df[numerical_cols].describe().to_string}
```

```
:Categorical Columns Summary
{'df[categorical_cols].describe().to_string() if categorical_cols else 'None'}
```

!PROJECT COMPLETED SUCCESSFULLY ✓

```
!All tasks completed and visualizations saved
('''')
```

```
(df.to_csv('cleaned_kidney_disease.csv', index=False
("print("\n✓ Cleaned dataset saved as: cleaned_kidney_disease.csv"
```

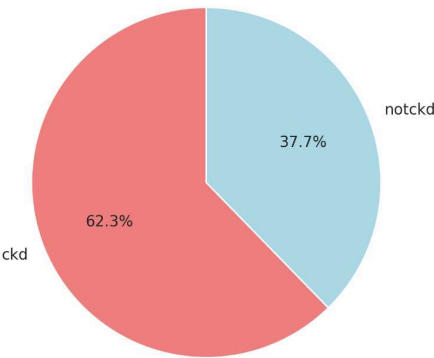
```
(print("\n" + "=" * 80
("):print("FILES GENERATED
(print("=" * 80
"""))print
:Python Code
(Project_1_Solution.py (this file ✓
```

```
:Visualizations
missing_values_comparison.png_01 ✓
correlation_heatmap.png_02 ✓
distributions.png_03 ✓
outliers_comparison.png_04 ✓
classification_distribution.png_05 ✓
```

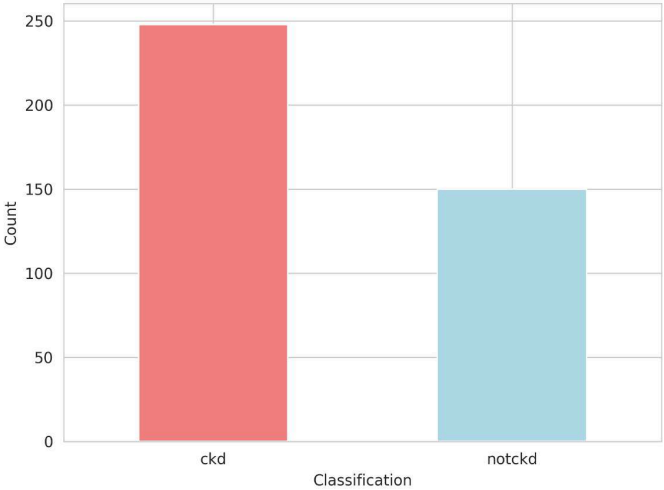
```
:Data
cleaned_kidney_disease.csv ✓
```

(

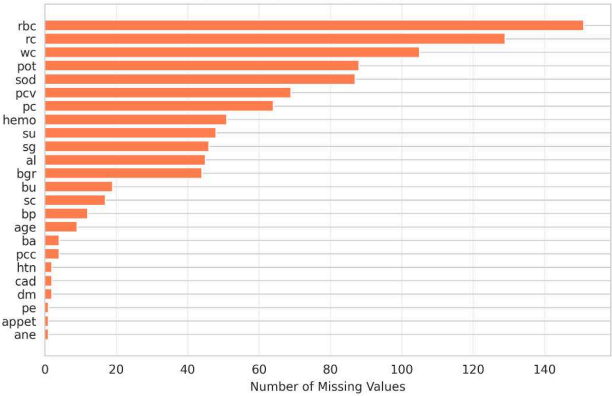
Classification Distribution (Pie Chart)



Classification Distribution (Bar Chart)



Missing Values BEFORE Cleaning



Missing Values AFTER Cleaning

✓ ALL MISSING VALUES
HAVE BEEN HANDLED!

