

Course Title: Practical Machine Learning and Data Exploration
Lecturer: Dr. Hager Saleh
Deadline Date: 17 July 2026
Hand-in Method: Online Submission via E-Learning.

Student Name: Emad Ahmad Al-Rubeh
ID: 232002310
Assignment: 1-1
Research Title: Project 1-1 (Kidney Disease)

Requirements:

1. Handle missing values.
2. Handle outliers.
3. Perform data visualization.

Project 1-1 (Kidney Disease)

Folder Contents:

- *Dataset Description.PNG*
 - *kidney_disease.csv*
 - *Project 1-1 (kidney Disease).ipynb*
 - *Projects.PNG*
-

Project Stages:

1. Data Cleaning & Pre-processing

- *Checking Missing Values in the Dataset*
- *Calculating the Percentage of Missing Values per Column*
- *Dropping Columns with a High Percentage of Missing Values*
- *Encoding Categorical Variables Using Label Encoder*
- *Handling Missing Values Based on Variable Type*

2. Data Visualization

- *Plotting a Correlation Heatmap*
 - *Correcting and Verifying Target Variable Classes*
 - *Refining Data and Counting Target Class Distribution*
 - *Plotting the Target Class Distribution Bar Chart*
 - *Plotting Average Hemoglobin Level by CKD Status*
 - *Splitting Features and Plotting Age Distribution*
 - *Binning Age and Creating a Comparison Table*
 - *Plotting Age Distribution by CKD Status Using FacetGrid*
 - *Plotting Appetite Status Distribution Pie Chart*
 - *Plotting Hemoglobin vs Age by CKD Status Using Line Plot*
 - *Plotting a Distribution Count of the Target Variable*
 - *Plotting Age vs Hemoglobin Scatter Plot by CKD Status*
 - *Plotting Hemoglobin Levels Boxplot by Classification*
-

➤ Displaying an Image:

```
from IPython.display import Image, display
display(Image(filename='projects.png'))
```



➤ Importing Essential Data Science Libraries:

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

➤ **Loading the Dataset and Displaying its Description:**

```
# Kidney disease project
df = pd.read_csv('kidney_disease.csv')
display(Image(filename='Dataset Description.png'))
```

Short Name	Full Name	Description
pc	Pus Cell	Urine pus cell (normal/abnormal)
pcc	Pus Cell Clumps	Presence of pus cell clumps in urine (present/notpresent)
ba	Bacteria	Presence of bacteria in urine (present/notpresent)
bgr	Blood Glucose Random	Random blood glucose in mgs/dl
bu	Blood Urea	Blood urea in mgs/dl
sc	Serum Creatinine	Serum creatinine in mgs/dl
sod	Sodium	Serum sodium in mEq/L
pot	Potassium	Serum potassium in mEq/L
hemo	Hemoglobin	Hemoglobin in gms
pcv	Packed Cell Volume	Packed cell volume (percentage or appropriate unit)
wc	White Blood Cell Count	White blood cell count in cells/cumm
rc	Red Blood Cell Count	Red blood cell count in millions/cmm
htn	Hypertension	Hypertension (yes/no)
dm	Diabetes Mellitus	Diabetes mellitus (yes/no)
cad	Coronary Artery Disease	Coronary artery disease (yes/no)
appet	Appetite	Appetite status (good/poor)
pe	Pedal Edema	Pedal edema (yes/no)
ane	Anemia	Anemia (yes/no)
classification	classification label	Indicates chronic kidney disease (ckd) or not (notckd)

➤ Previewing the First (5) Rows of the Dataset

```
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

➤ Checking Data Types of Features

```
# Getting features type  
df.dtypes
```

```
id                int64  
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             object  
wc              object  
rc              object  
htn             object  
dm              object  
cad             object  
appet           object  
pe              object  
ane             object  
classification   object  
dtype: object
```

➤ Listing Dataset Columns

```
# Print columns
df.columns

Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
       'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm',
       'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

❖ Data Cleaning & Pre-processing

➤ Checking Missing Values in the Dataset

```
# Getting the count of missing values
df.isnull().sum()

id          0
age         9
bp         12
sg         47
al         46
su         49
rbc        152
pc         65
pcc         4
ba          4
bgr        44
bu         19
sc         17
sod        87
pot        88
hemo       52
pcv        70
wc        105
rc        130
htn         2
dm          2
cad         2
appet       1
pe          1
ane         1
classification  0
dtype: int64
```

➤ Calculating the Percentage of Missing Values per Column

```
# Load the dataset
df = pd.read_csv('kidney_disease.csv')

# Calculate the percentage of missing values per column
missing_percentage = (df.isnull().sum() / len(df)) * 100

# Display the result
print(missing_percentage)
```

```
id          0.00
age         2.25
bp          3.00
sg         11.75
al         11.50
su         12.25
rbc        38.00
pc         16.25
pcc         1.00
ba          1.00
bgr        11.00
bu          4.75
sc          4.25
sod        21.75
pot        22.00
hemo       13.00
pcv        17.50
wc         26.25
rc         32.50
htn         0.50
dm          0.50
cad         0.50
appet       0.25
pe          0.25
ane         0.25
classification 0.00
dtype: float64
```

➤ Dropping Columns with High Percentage (30%) of Missing Values

```
# Deleting columns that have 30% missing values
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"\nRemaining columns: {df.columns}")
```

```
Columns dropped: ['rbc', 'rc']
```

```
Remaining columns: Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'pc', 'pcc',
'ba', 'bgr', 'bu',
      'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'htn', 'dm', 'cad', 'appet',
      'pe', 'ane', 'classification'],
      dtype='object')
```

➤ Encoding Categorical Variables Using Label Encoder

```
# Pre-processing on dataset
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

	id	age	bp	sg	al	su	pc	pcc	ba	bgr	...	hemo	pcv	wc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	1	0	0	121.0	...	15.4	32	72	1	4	1	0	0	0	0
1	1	7.0	50.0	1.020	4.0	0.0	1	0	0	NaN	...	11.3	26	56	0	3	1	0	0	0	0
2	2	62.0	80.0	1.010	2.0	3.0	1	0	0	423.0	...	9.6	19	70	0	4	1	1	0	1	0
3	3	48.0	70.0	1.005	4.0	0.0	0	1	0	117.0	...	11.2	20	62	1	3	1	1	1	1	0
4	4	51.0	80.0	1.010	2.0	0.0	1	0	0	106.0	...	11.6	23	68	0	3	1	0	0	0	0

5 rows × 24 columns

➤ Handling Missing Values Based on Variable Type

```
# Filling missing values based on feature type:
# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)

# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()
```

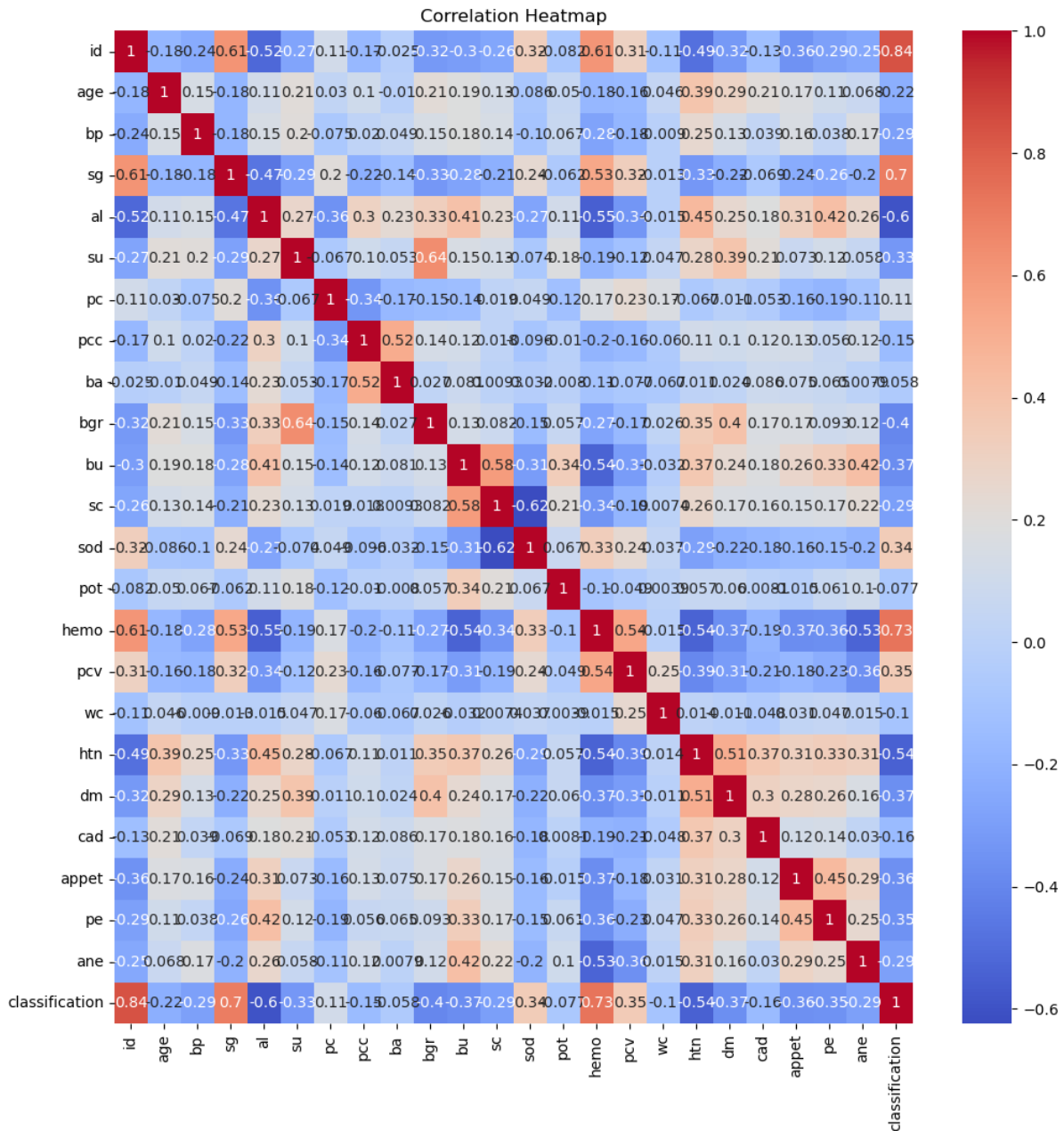
```
id          0
age         0
bp          0
sg          0
al          0
su          0
pc          0
pcc         0
ba          0
bgr         0
bu          0
sc          0
sod         0
pot         0
hemo        0
pcv         0
wc          0
htn         0
dm          0
cad         0
appet       0
pe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

❖ Data Visualization

➤ Plotting a Correlation Heatmap

```
# Data Visualization
# Correlation heatmap
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



➤ Correcting and Verifying Target Variable Classes

```
import pandas as pd

# Load the dataset
df = pd.read_csv('kidney_disease.csv')

# Correct the 'ckd\t' entry to 'ckd' as requested
df['classification'] = df['classification'].replace('ckd\t', 'ckd')

# Get the list of unique values
unique_classification_list = df['classification'].unique().tolist()

# Print the result
print(unique_classification_list)
```

```
['ckd', 'notckd']
```

➤ Refining Data and Counting Target Class Distribution

```
# Handle Missing Values (Drop columns with > 30% missing data)
missing_percentage = df.isnull().mean() * 100
threshold = 30
columns_to_drop = missing_percentage[missing_percentage > threshold].index
df = df.drop(columns=columns_to_drop, axis=1)

# Drop the 'id' column
if 'id' in df.columns:
    df = df.drop(columns=['id'])

# Fill remaining missing values
# Ensures the dataset is complete before encoding
df = df.ffill().bfill()

# Pre-processing (Encoding)
# Convert all categorical object columns into numerical labels
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])

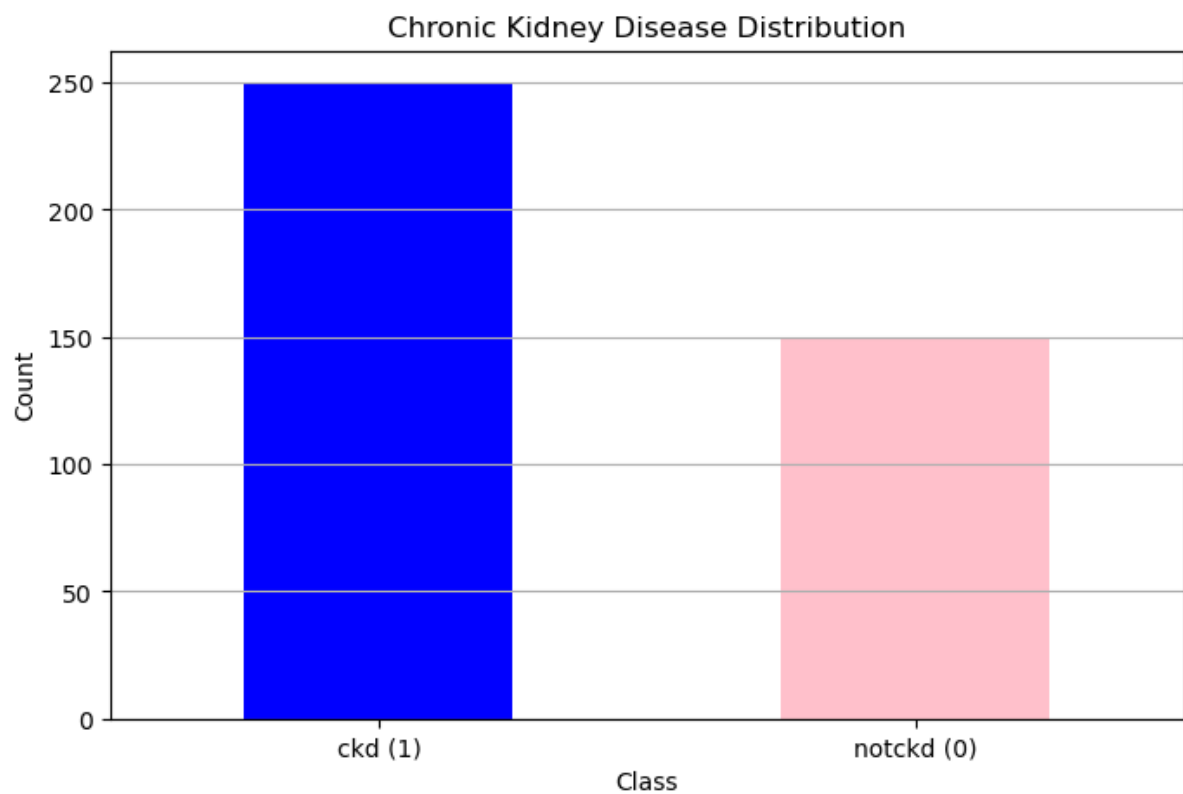
# Print the resulting counts for classification
df['classification'].value_counts()
```

```
classification
0      250
1      150
Name: count, dtype: int64
```

➤ Plotting the Target Class Distribution Bar Chart

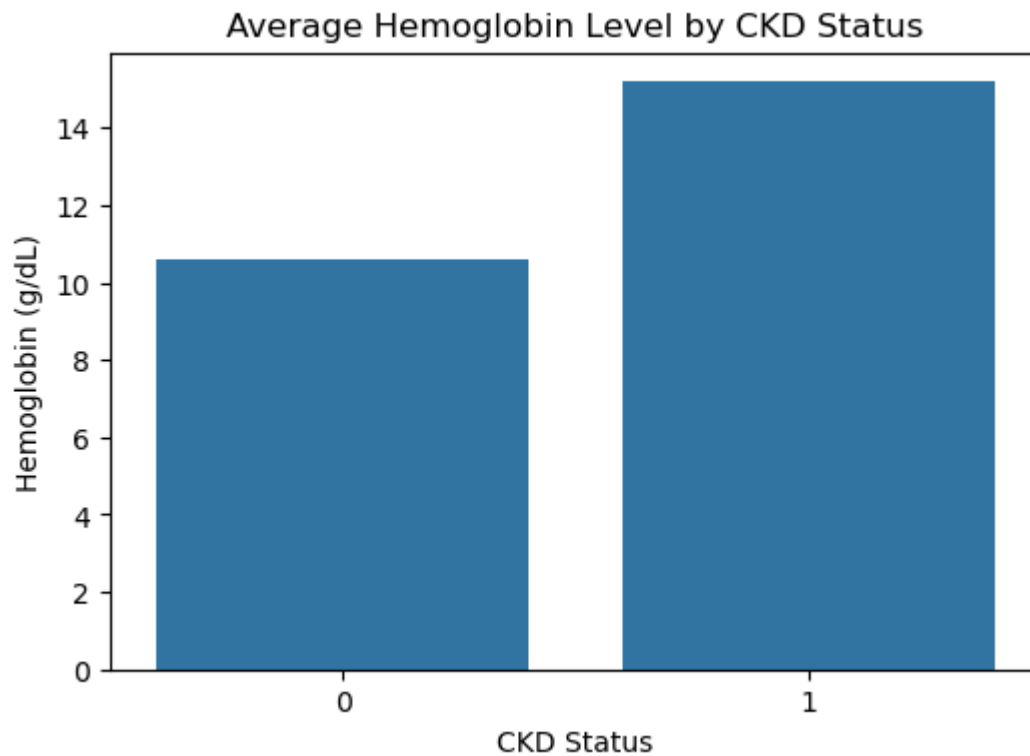
Create a bar chart

```
plt.figure(figsize=(8, 5))
df['classification'].value_counts().plot(kind='bar', color=['blue',
'pink'])
plt.xticks([0, 1], ['ckd (1)' , 'notckd (0)'], rotation=0)
plt.title('Chronic Kidney Disease Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



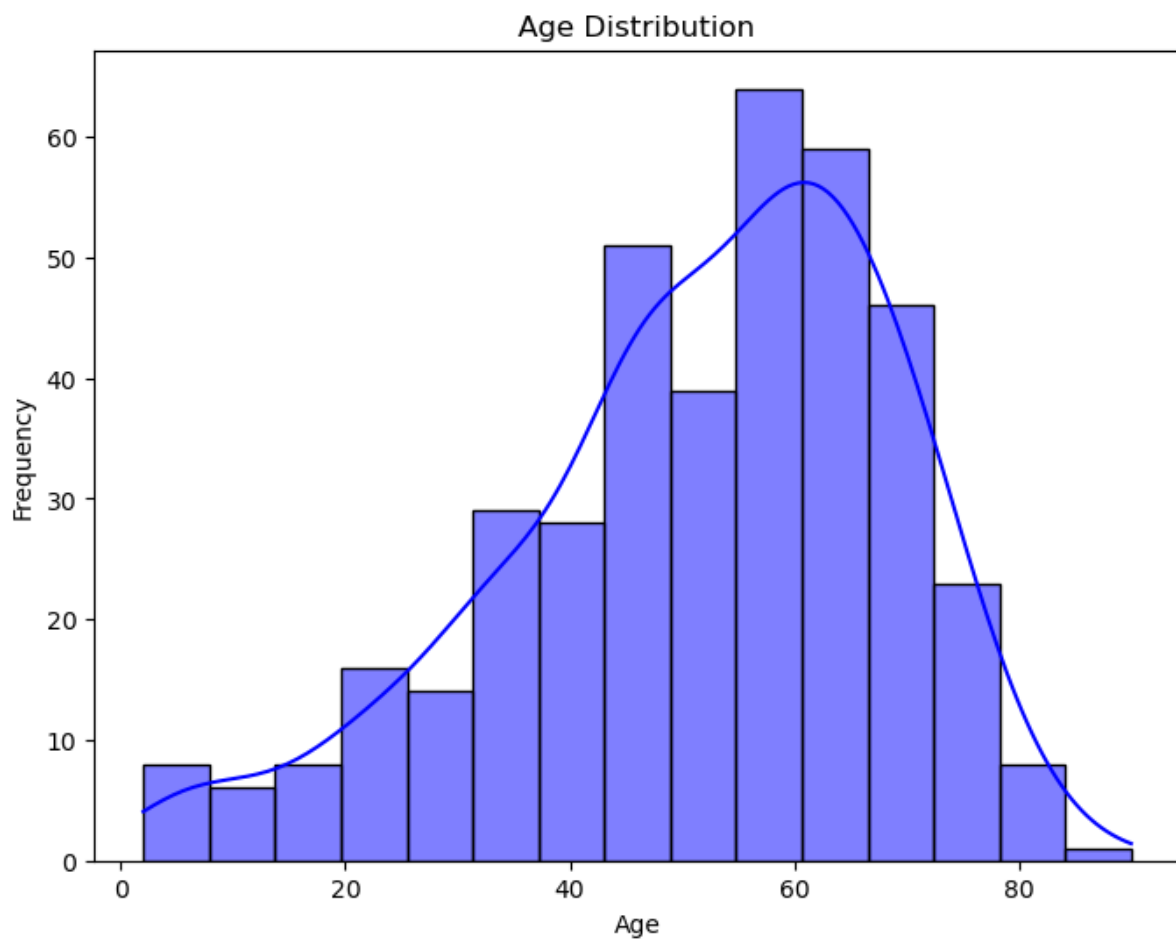
➤ Plotting Average Hemoglobin Level by CKD Status

```
# Example 2: Average Hemoglobin by CKD Status
import seaborn as sns
plt.figure(figsize=(6,4))
sns.barplot(x='classification', y='hemo', data=df, estimator='mean',
ci=None)
plt.title("Average Hemoglobin Level by CKD Status")
plt.xlabel("CKD Status")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```



➤ Splitting Features and Plotting Age Distribution

```
cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
# 1. Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], bins=15, kde=True, color='blue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



➤ Binning Age and Creating a Comparison Table

```
import pandas as pd

# Dividing ages into 15 categories to match the chart
bins = range(0, 92, 6)
df['age_group'] = pd.cut(df['age'], bins=bins)

# Create a comparison table between (ckd & notckd) for each age group.
comparison_table = df.groupby(['age_group',
                                'classification']).size().unstack(fill_value=0)

# Print the resulting counts for classification ages (ckd & notckd)
print(comparison_table)

# Calculate the Min & Max ages
min_age = df['age'].min()
max_age = df['age'].max()

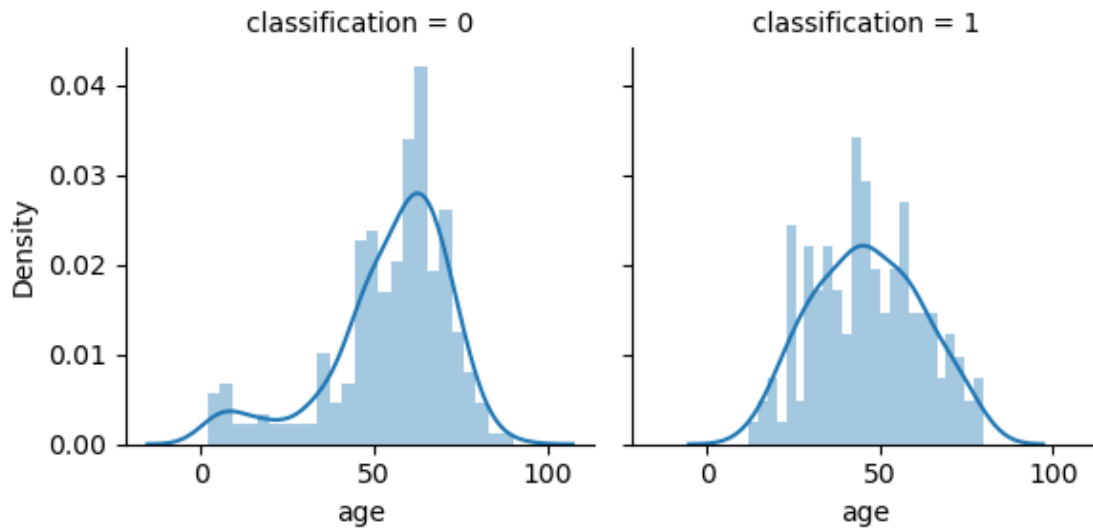
print(f"\nMin Age: {min_age}")
print(f"Max Age: {max_age}")
```

classification	0	1
age_group		
(0, 6]	7	0
(6, 12]	6	1
(12, 18]	4	2
(18, 24]	4	11
(24, 30]	3	14
(30, 36]	10	16
(36, 42]	9	16
(42, 48]	31	26
(48, 54]	27	12
(54, 60]	41	23
(60, 66]	47	12
(66, 72]	37	9
(72, 78]	18	5
(78, 84]	5	3
(84, 90]	1	0

```
Min Age: 2.0
Max Age: 90.0
```

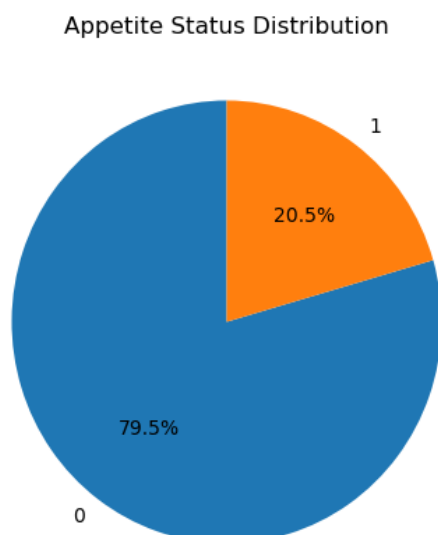
➤ Plotting Age Distribution by CKD Status Using FacetGrid

```
g = sns.FacetGrid(df, col="classification")
g.map(sns.distplot, "age", bins=25)
plt.show()
```



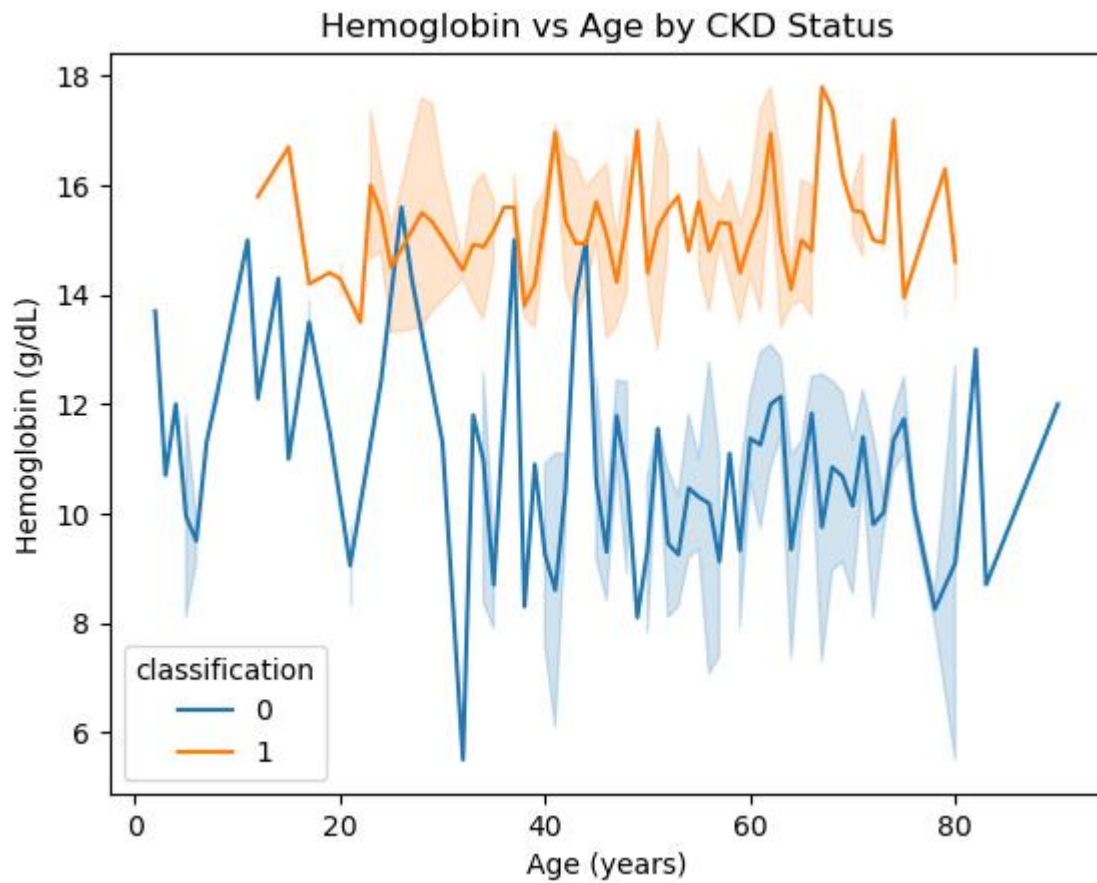
➤ Plotting Appetite Status Distribution Pie Chart

```
appet_counts = df['appet'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%',
startangle=90)
plt.title("Appetite Status Distribution")
plt.show()
```



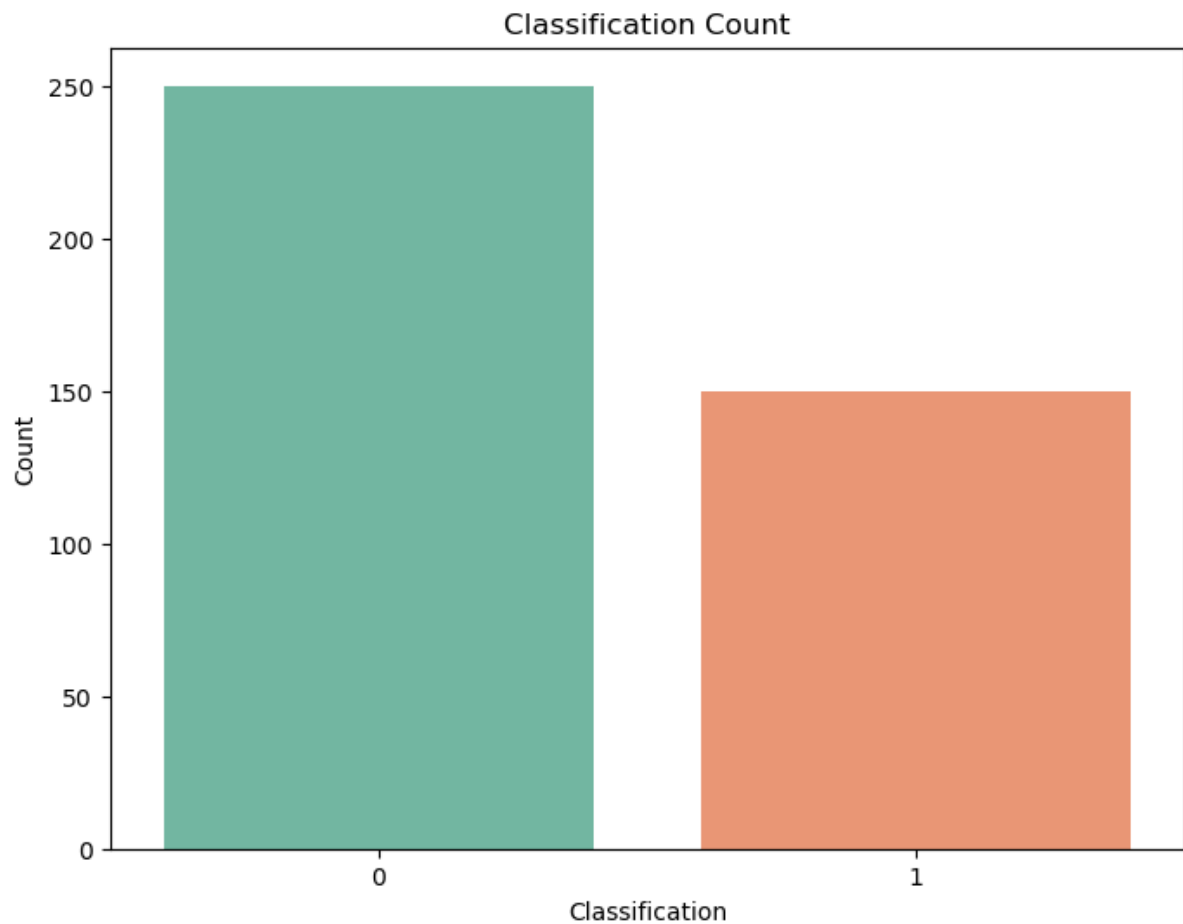
➤ Plotting Hemoglobin vs Age by CKD Status Using Line Plot

```
sns.lineplot(x='age', y='hemo', hue='classification', data=df)
plt.title("Hemoglobin vs Age by CKD Status")
plt.xlabel("Age (years)")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```



➤ Plotting a Distribution Count of the Target Variable

```
# 3. Countplot for Classification
plt.figure(figsize=(8, 6))
sns.countplot(x='classification', data=df, palette='Set2')
plt.title('Classification Count')
plt.xlabel('Classification')
plt.ylabel('Count')
plt.show()
```

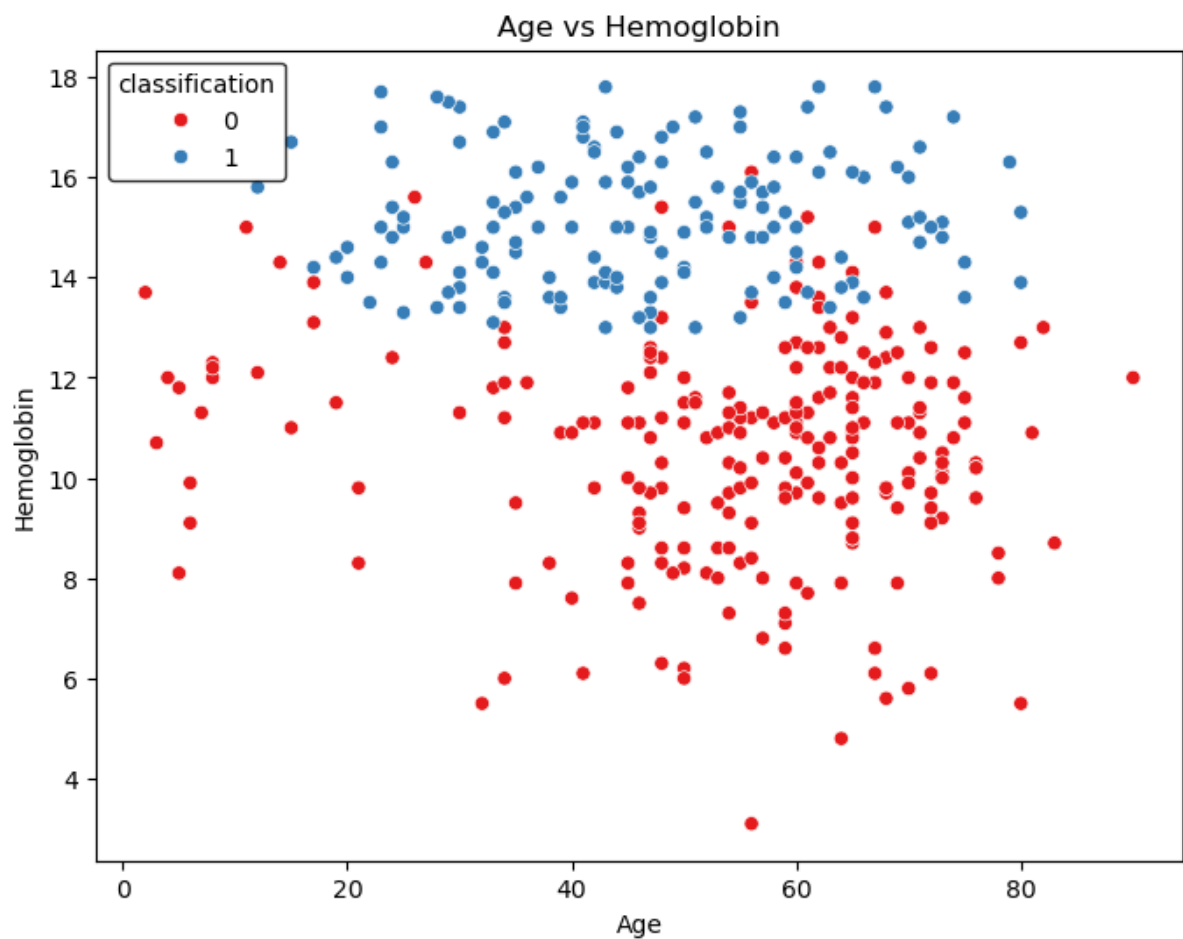


➤ Plotting Age vs Hemoglobin Scatter Plot by CKD Status

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='hemo', data=df, hue='classification',
               palette='Set1')

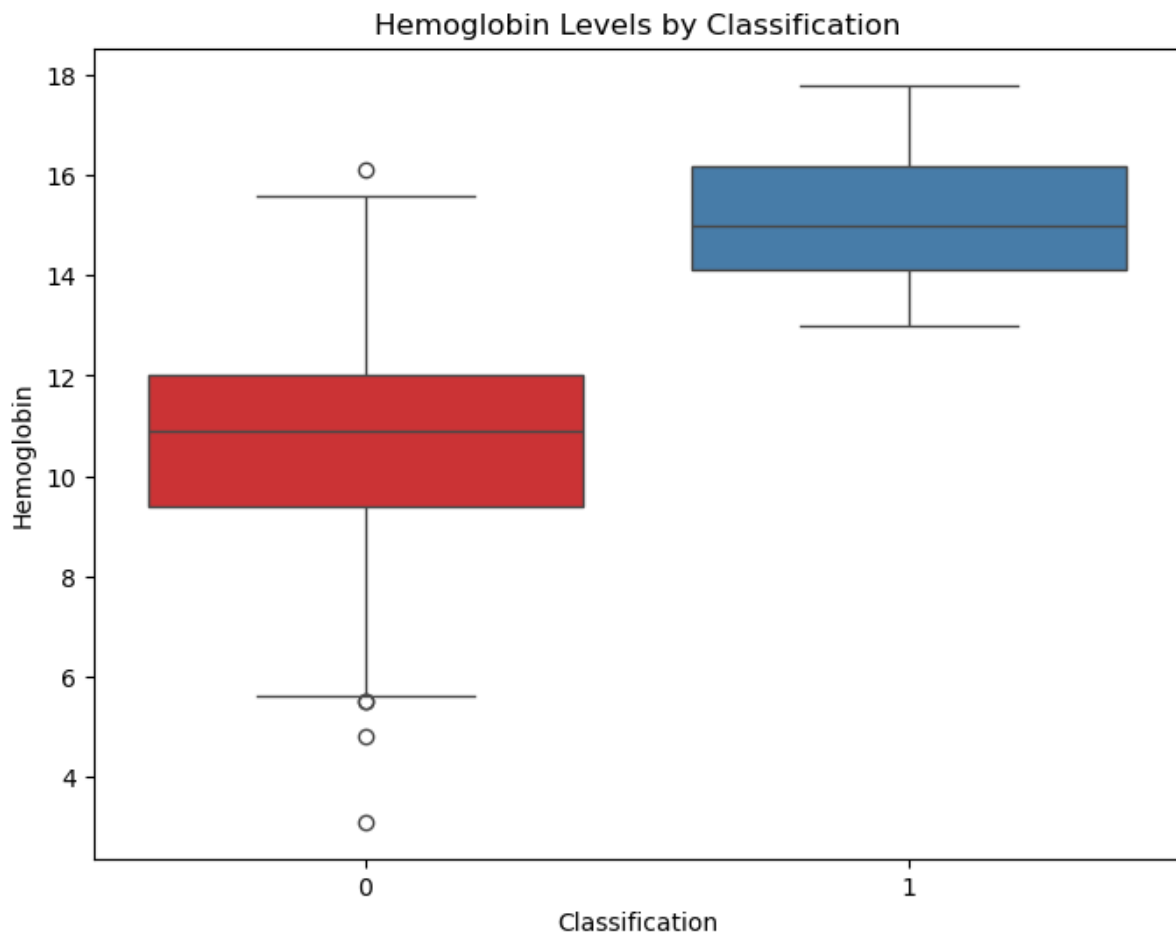
# Showing frame around the classification
plt.legend(title='classification', frameon=True, edgecolor='black')

plt.title('Age vs Hemoglobin')
plt.xlabel('Age')
plt.ylabel('Hemoglobin')
plt.show()
```



➤ Plotting Hemoglobin Levels Boxplot by Classification

```
# Boxplot for Hemoglobin by Classification
plt.figure(figsize=(8, 6))
sns.boxplot(x='classification', y='hemo', data=df, palette='Set1')
plt.title('Hemoglobin Levels by Classification')
plt.xlabel('Classification')
plt.ylabel('Hemoglobin')
plt.show()
```



The End