

In [114...



In [109...

```
import pandas as pd
import numpy as np
```

The data link <https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease>

In [110...

```
# Load your dataset into a DataFrame
df = pd.read_csv('kidney_disease.csv')
# Preview the first 5 rows
df.head()
```

Out[110...

	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

Getting features type

In [111...

```
df.dtypes
```

```
Out[111...  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
           dtype: object
```

```
In [112... df.describe()
```

```
Out[112...      age      bp      sg      al      su      bgr      bu
count  389.000000  386.000000  352.000000  353.000000  350.000000  354.000000  379.000000
mean    51.395887   76.502591   1.017429   1.014164   0.451429  147.943503   57.512401
std     17.168437   13.692210   0.005711   1.353582   1.100500   79.469325   50.592153
min      2.000000   50.000000   1.005000   0.000000   0.000000   22.000000    1.500000
25%     42.000000   70.000000   1.010000   0.000000   0.000000   99.000000   27.000000
50%     54.000000   80.000000   1.020000   0.000000   0.000000  121.000000   42.000000
75%     64.000000   80.000000   1.020000   2.000000   0.000000  162.750000   66.000000
max     90.000000  180.000000   1.025000   5.000000   5.000000  490.000000  391.000000
```

Getting the count of missing values

```
In [92]: df.isnull().sum()
```

```
Out[92]: age          9
         bp          12
         sg          46
         al          45
         su          48
         rbc        151
         pc          64
         pcc         4
         ba          4
         bgr        44
         bu          19
         sc          17
         sod         87
         pot         88
         hemo        51
         pcv         69
         wc         105
         rc         129
         htn         2
         dm          2
         cad         2
         appet       1
         pe          1
         ane         1
         classification 0
         dtype: int64
```

Deleting columns that have 30% missing values

```
In [93]: 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}")
```

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

Remaining columns: Index(['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')

Pre-processing on dataset

```
In [94]: 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()
```

```
Out[94]:
```

	age	bp	sg	al	su	pc	pcc	ba	bgr	bu	...	hemo	pcv	wc	htn	dm	cad
0	48.0	80.0	1.020	1.0	0.0	1	0	0	121.0	36.0	...	15.4	32	72	1	4	1
1	7.0	50.0	1.020	4.0	0.0	1	0	0	NaN	18.0	...	11.3	26	56	0	3	1
2	62.0	80.0	1.010	2.0	3.0	1	0	0	423.0	53.0	...	9.6	19	70	0	4	1
3	48.0	70.0	1.005	4.0	0.0	0	1	0	117.0	56.0	...	11.2	20	62	1	3	1
4	51.0	80.0	1.010	2.0	0.0	1	0	0	106.0	26.0	...	11.6	23	68	0	3	1

5 rows × 23 columns

Filling missing values based on feature type

```
In [95]: # 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)
```

```
In [96]: # 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)
```

```
In [97]: df.isnull().sum()
```

```
Out[97]: 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
```

```
In [98]: df
```

Out[98]:

	age	bp	sg	al	su	pc	pcc	ba	bgr	bu	...	hemo	pcv	wc	htn	d
0	48.0	80.0	1.020	1.0	0.0	1	0	0	121.000000	36.0	...	15.4	32	72	1	
1	7.0	50.0	1.020	4.0	0.0	1	0	0	147.943503	18.0	...	11.3	26	56	0	
2	62.0	80.0	1.010	2.0	3.0	1	0	0	423.000000	53.0	...	9.6	19	70	0	
3	48.0	70.0	1.005	4.0	0.0	0	1	0	117.000000	56.0	...	11.2	20	62	1	
4	51.0	80.0	1.010	2.0	0.0	1	0	0	106.000000	26.0	...	11.6	23	68	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
393	55.0	80.0	1.020	0.0	0.0	1	0	0	140.000000	49.0	...	15.7	35	62	0	
394	42.0	70.0	1.025	0.0	0.0	1	0	0	75.000000	31.0	...	16.5	42	72	0	
395	12.0	80.0	1.020	0.0	0.0	1	0	0	100.000000	26.0	...	15.8	37	61	0	
396	17.0	60.0	1.025	0.0	0.0	1	0	0	114.000000	50.0	...	14.2	39	67	0	
397	58.0	80.0	1.025	0.0	0.0	1	0	0	131.000000	18.0	...	15.8	41	63	0	

398 rows × 23 columns

Detect outliers

```
In [99]: 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 df.columns:
    df[col] = replace_outliers_with_nan(df[col])
```

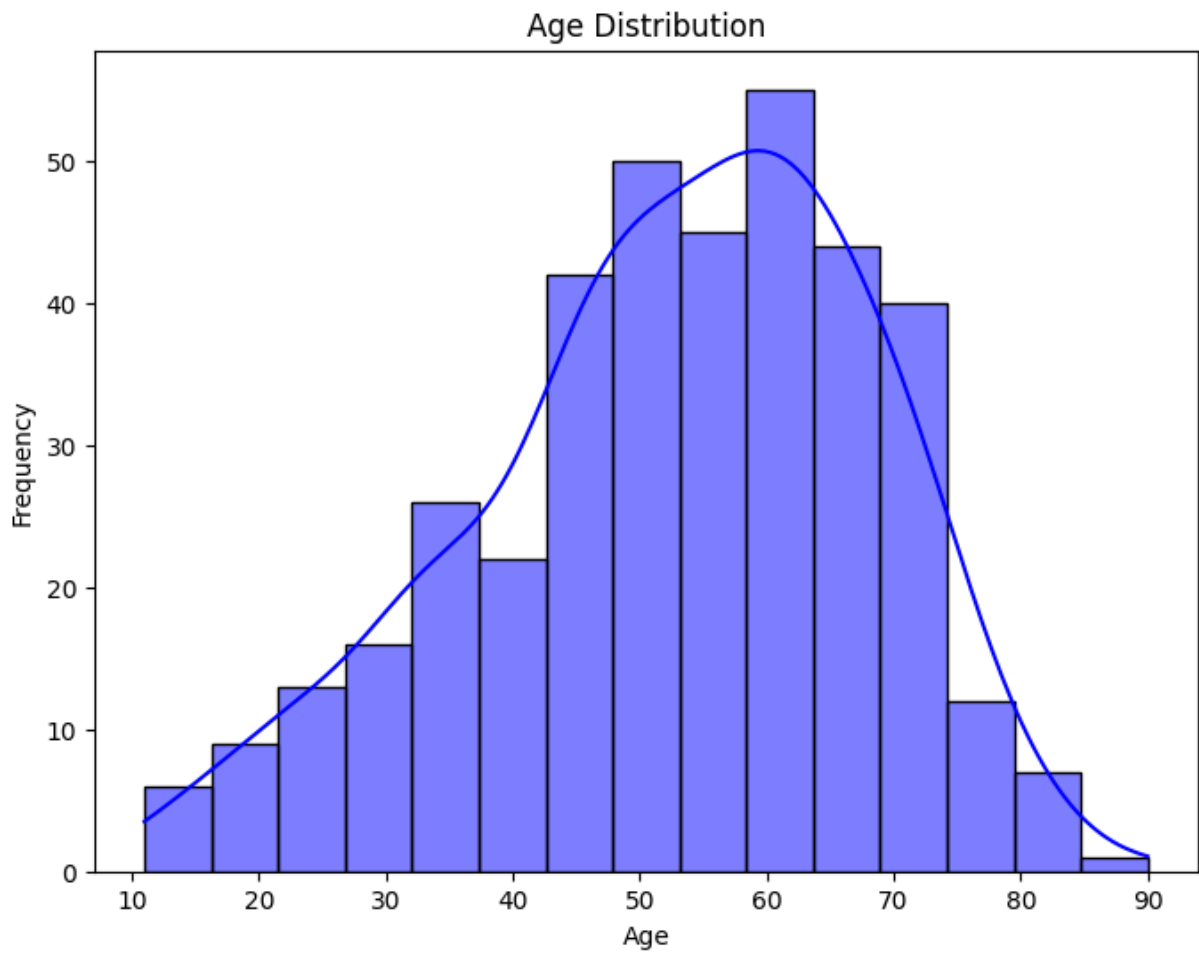
```
In [100... null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

Sum of null values in each column:

```
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 Visualizatiion

```
In [101... # 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()
```

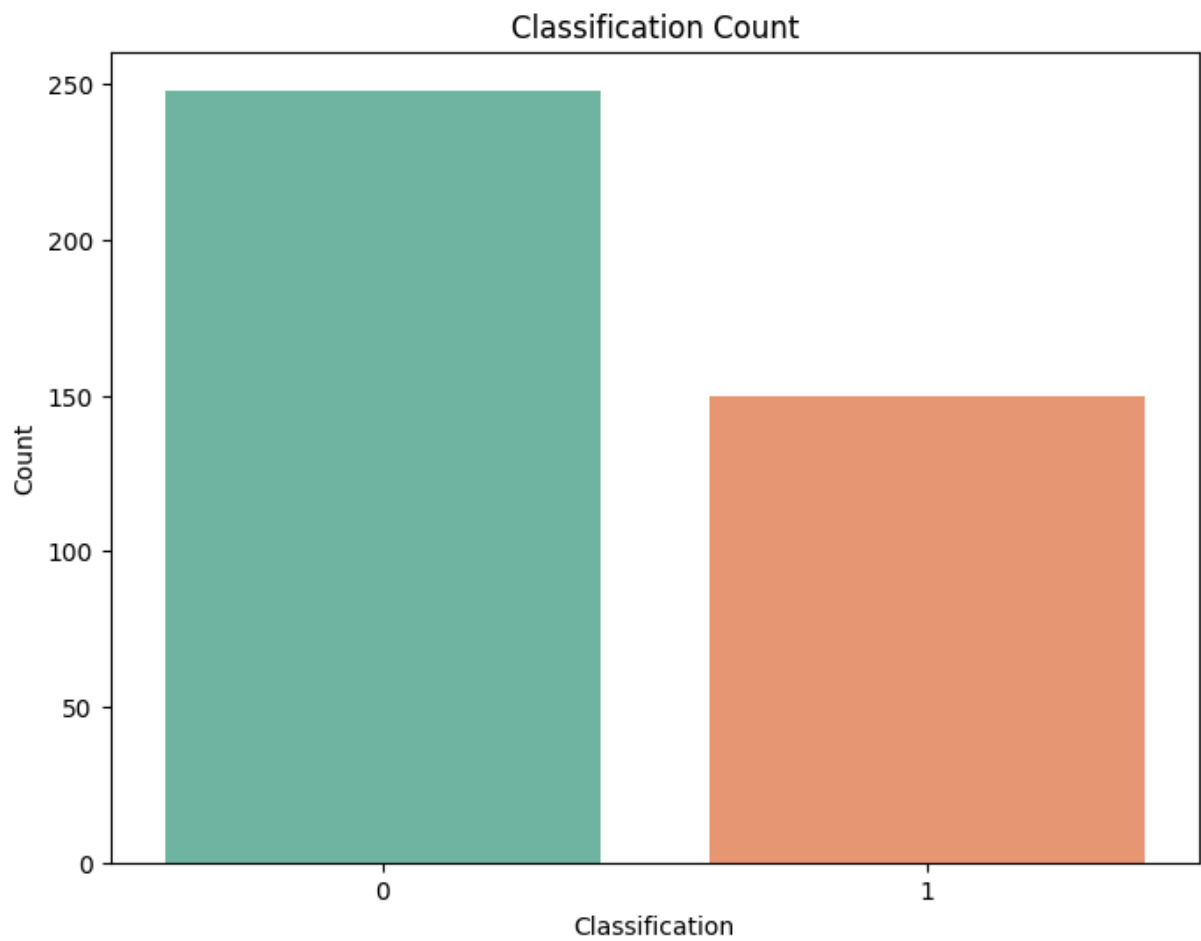


```
In [103... # 3. Countplot for Classification
plt.figure(figsize=(8, 6))
sns.countplot(x='classification', data=df, palette='Set2')
plt.title('Classification Count')
plt.xlabel('Classification')
plt.ylabel('Count')
plt.show()
```

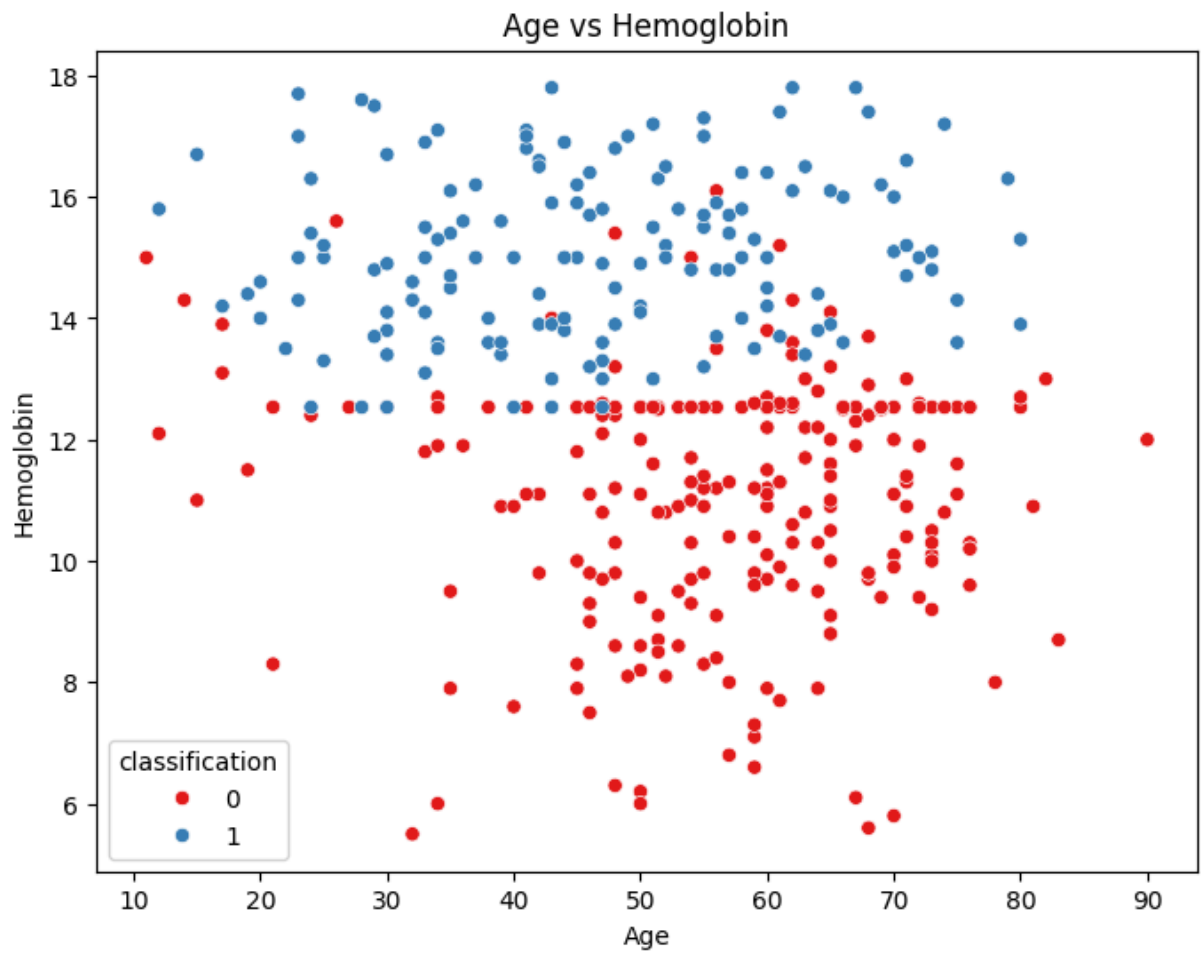
C:\Users\hager\AppData\Local\Temp\ipykernel_29692\3237698328.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(x='classification', data=df, palette='Set2')
```



```
In [105... plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='hemo', data=df, hue='classification', palette='Set1')
plt.title('Age vs Hemoglobin')
plt.xlabel('Age')
plt.ylabel('Hemoglobin')
plt.show()
```



```
In [106... # Scatter plot for Age vs Blood Pressure
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='bp', data=df, hue='classification', palette='Set2')
plt.title('Age vs Blood Pressure')
plt.xlabel('Age')
plt.ylabel('Blood Pressure')
plt.show()
```



In [107...

```
# 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()
```

C:\Users\hager\AppData\Local\Temp\ipykernel_29692\1790484353.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.boxplot(x='classification', y='hemo', data=df, palette='Set1')
```

