```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

```
#Reading Csv File
```

```
df = pd.read_csv('Kidney_Disease.csv')
```

```
#Displaying Top Rows
```

```
df.head()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300

5 rows × 26 columns

```
#Dropping Id column
```

```
df.drop('id', axis =1 , inplace= True)
```

```
# List the Columns
```

```
df.columns
```

```
Index(['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')
```

```
#Rename the Columns for better Understanding
```

```
df.columns = ['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cell',
              'pus_cell_clumps', 'bacteria', 'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium',
              'potassium', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count',
              'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite', 'peda_edema',
              'aanemia', 'class']
```

```
# Statistical Data Table
```

```
df.describe()
```

	age	blood_pressure	specific_gravity	albumin	sugar	blood_glu
count	391.000000	388.000000	353.000000	354.000000	351.000000	
mean	51.483376	76.469072	1.017408	1.016949	0.450142	
std	17.169714	13.683637	0.005717	1.352679	1.099191	
min	2.000000	50.000000	1.005000	0.000000	0.000000	
25%	42.000000	70.000000	1.010000	0.000000	0.000000	
50%	55.000000	80.000000	1.020000	0.000000	0.000000	
75%	64.500000	80.000000	1.020000	2.000000	0.000000	
max	90.000000	180.000000	1.025000	5.000000	5.000000	

```
#Checking for Null and Data Type of Each Column
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      391 non-null    float64
 1   blood_pressure           388 non-null    float64
 2   specific_gravity         353 non-null    float64
 3   albumin                  354 non-null    float64
 4   sugar                    351 non-null    float64
 5   red_blood_cells          248 non-null    object
 6   pus_cell                 335 non-null    object
 7   pus_cell_clumps          396 non-null    object
 8   bacteria                 396 non-null    object
 9   blood_glucose_random     356 non-null    float64
 10  blood_urea               381 non-null    float64
 11  serum_creatinine         383 non-null    float64
 12  sodium                   313 non-null    float64
 13  potassium                312 non-null    float64
 14  haemoglobin              348 non-null    float64
 15  packed_cell_volume       330 non-null    object
 16  white_blood_cell_count   295 non-null    object
 17  red_blood_cell_count     270 non-null    object
 18  hypertension             398 non-null    object
 19  diabetes_mellitus        398 non-null    object
 20  coronary_artery_disease  398 non-null    object
 21  appetite                 399 non-null    object
 22  peda_edema               399 non-null    object
 23  aanemia                  399 non-null    object
 24  class                    400 non-null    object
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

```
#Seprating Categorical and Numerical Columns


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"]


#Printing Unique Values of each columns


for col in cat_cols:
    print(f"{col} has {df[col].unique()} values \n")
```

```
red_blood_cells has [nan 'normal' 'abnormal'] values

pus_cell has ['normal' 'abnormal' nan] values

pus_cell_clumps has ['notpresent' 'present' nan] values

bacteria has ['notpresent' 'present' nan] values

packed_cell_volume has ['44' '38' '31' '32' '35' '39' '36' '33' '29' '28' nan '16' '24' '37' '30'
 '34' '40' '45' '27' '48' '\t?' '52' '14' '22' '18' '42' '17' '46' '23'
 '19' '25' '41' '26' '15' '21' '43' '20' '\t43' '47' '9' '49' '50' '53'
 '51' '54'] values

white_blood_cell_count has ['7800' '6000' '7500' '6700' '7300' nan '6900' '9600' '12100' '4500'
 '12200' '11000' '3800' '11400' '5300' '9200' '6200' '8300' '8400' '10300'
 '9800' '9100' '7900' '6400' '8600' '18900' '21600' '4300' '8500' '11300'
 '7200' '7700' '14600' '6300' '\t6200' '7100' '11800' '9400' '5500' '5800'
 '13200' '12500' '5600' '7000' '11900' '10400' '10700' '12700' '6800'
 '6500' '13600' '10200' '9000' '14900' '8200' '15200' '5000' '16300'
 '12400' '\t8400' '10500' '4200' '4700' '10900' '8100' '9500' '2200'
 '12800' '11200' '19100' '\t?' '12300' '16700' '2600' '26400' '8800'
 '7400' '4900' '8000' '12000' '15700' '4100' '5700' '11500' '5400' '10800'
 '9900' '5200' '5900' '9300' '9700' '5100' '6600'] values

red_blood_cell_count has ['5.2' nan '3.9' '4.6' '4.4' '5' '4.0' '3.7' '3.8' '3.4' '2.6' '2.8' '4.3'
 '3.2' '3.6' '4' '4.1' '4.9' '2.5' '4.2' '4.5' '3.1' '4.7' '3.5' '6.0'
 '5.0' '2.1' '5.6' '2.3' '2.9' '2.7' '8.0' '3.3' '3.0' '3' '2.4' '4.8'
 '\t?' '5.4' '6.1' '6.2' '6.3' '5.1' '5.8' '5.5' '5.3' '6.4' '5.7' '5.9'
 '6.5'] values

hypertension has ['yes' 'no' nan] values

diabetes_mellitus has ['yes' 'no' ' yes' '\tno' '\tyes' nan] values

coronary_artery_disease has ['no' 'yes' '\tno' nan] values

appetite has ['good' 'poor' nan] values

peda_edema has ['no' 'yes' nan] values
```

```
aanemia has ['no' 'yes' nan] values

class has ['ckd' 'ckd\t' 'notckd'] values
```

```
#Replace Unwanted Values

df['diabetes_mellitus'].replace(to_replace = {' yes':'yes', '\tyes': 'yes', '\tno': 'no'}, inplace = True)

df['coronary_artery_disease'] = df['coronary_artery_disease'].replace(to_replace = '\tno', value = 'no')

df['class'] = df['class'].replace(to_replace = 'ckd\t', value = 'ckd')

df['class'] = df['class'].map({'ckd': 0, 'notckd': 1}).astype(int)

df['packed_cell_volume'].replace(to_replace = {'\t?':'nan', '\t43': 43}, inplace = True)

df['white_blood_cell_count'].replace(to_replace = {'\t?':'nan', '\t6200':6200,'\t8400':8400}, inplace = True)

df['red_blood_cell_count'].replace(to_replace = {'\t?':'nan', '\t': ''}, inplace = True)

#Checking for Unwanted Values

for col in cat_cols:
    print(f"{col} has {df[col].unique()} values \n")
```

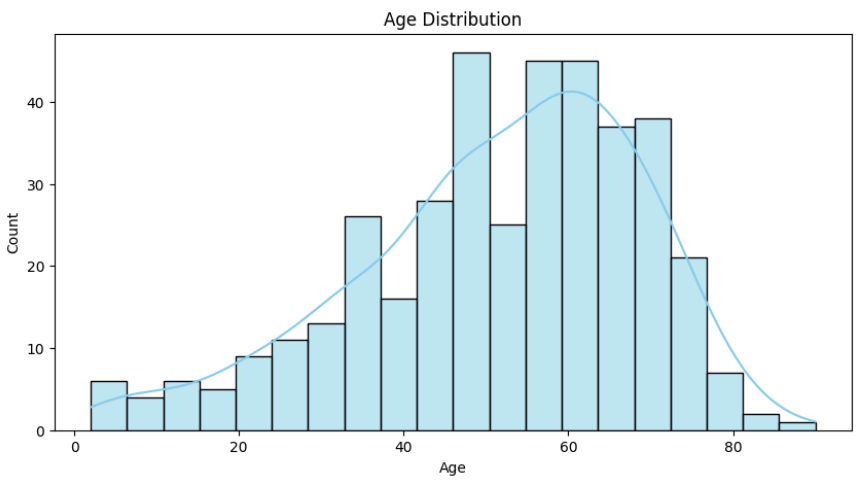
```
red_blood_cells has [nan 'normal' 'abnormal'] values

pus_cell has ['normal' 'abnormal' nan] values

pus_cell_clumps has ['notpresent' 'present' nan] values

bacteria has ['notpresent' 'present' nan] values

packed_cell_volume has ['44' '38' '31' '32' '35' '39' '36' '33' '29' '28' nan '16' '24' '37' '30'
 '34' '40' '45' '27' '48' 'nan' '52' '14' '22' '18' '42' '17' '46' '23'
 '19' '25' '41' '26' '15' '21' '43' '20' 43 '47' '9' '49' '50' '53' '51'
 '54'] values

white_blood_cell_count has ['7800' '6000' '7500' '6700' '7300' nan '6900' '9600' '12100' '4500'
 '12200' '11000' '3800' '11400' '5300' '9200' '6200' '8300' '8400' '10300'
 '9800' '9100' '7900' '6400' '8600' '18900' '21600' '4300' '8500' '11300'
 '7200' '7700' '14600' '6300' 6200 '7100' '11800' '9400' '5500' '5800'
 '13200' '12500' '5600' '7000' '11900' '10400' '10700' '12700' '6800'
 '6500' '13600' '10200' '9000' '14900' '8200' '15200' '5000' '16300'
 '12400' 8400 '10500' '4200' '4700' '10900' '8100' '9500' '2200' '12800'
 '11200' '19100' 'nan' '12300' '16700' '2600' '26400' '8800' '7400' '4900'
 '8000' '12000' '15700' '4100' '5700' '11500' '5400' '10800' '9900' '5200'
 '5900' '9300' '9700' '5100' '6600'] values

red_blood_cell_count has ['5.2' nan '3.9' '4.6' '4.4' '5' '4.0' '3.7' '3.8' '3.4' '2.6' '2.8' '4.3'
 '3.2' '3.6' '4' '4.1' '4.9' '2.5' '4.2' '4.5' '3.1' '4.7' '3.5' '6.0'
 '5.0' '2.1' '5.6' '2.3' '2.9' '2.7' '8.0' '3.3' '3.0' '3' '2.4' '4.8'
 'nan' '5.4' '6.1' '6.2' '6.3' '5.1' '5.8' '5.5' '5.3' '6.4' '5.7' '5.9'
 '6.5'] values

hypertension has ['yes' 'no' nan] values

diabetes_mellitus has ['yes' 'no' nan] values

coronary_artery_disease has ['no' 'yes' nan] values

appetite has ['good' 'poor' nan] values

peda_edema has ['no' 'yes' nan] values

aanemia has ['no' 'yes' nan] values

class has [0 1] values
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      391 non-null    float64
 1   blood_pressure           388 non-null    float64
```

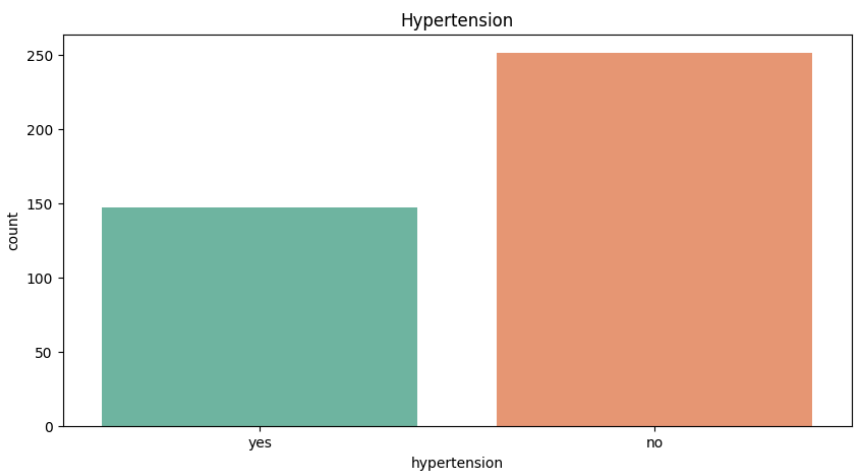
```
 2   specific_gravity         353 non-null    float64
 3   albumin                  354 non-null    float64
 4   sugar                    351 non-null    float64
 5   red_blood_cells          248 non-null    object
 6   pus_cell                 335 non-null    object
 7   pus_cell_clumps          396 non-null    object
 8   bacteria                 396 non-null    object
 9   blood_glucose_random     356 non-null    float64
 10  blood_urea               381 non-null    float64
 11  serum_creatinine         383 non-null    float64
 12  sodium                   313 non-null    float64
 13  potassium                312 non-null    float64
 14  haemoglobin              348 non-null    float64
 15  packed_cell_volume       330 non-null    object
 16  white_blood_cell_count   295 non-null    object
 17  red_blood_cell_count     270 non-null    object
 18  hypertension             398 non-null    object
 19  diabetes_mellitus        398 non-null    object
 20  coronary_artery_disease  398 non-null    object
 21  appetite                 399 non-null    object
 22  peda_edema               399 non-null    object
 23  aanemia                  399 non-null    object
 24  class                    400 non-null    int64
dtypes: float64(11), int64(1), object(13)
memory usage: 78.2+ KB
```

```
#Changing Data Type


df['packed_cell_volume'] = pd.to_numeric(df['packed_cell_volume'], errors = 'coerce')


df['white_blood_cell_count'] = pd.to_numeric(df['white_blood_cell_count'], errors = 'coerce')

df['red_blood_cell_count'] = pd.to_numeric(df['red_blood_cell_count'], errors = 'coerce')


df['class'] = pd.to_numeric(df['class'], errors = 'coerce')


                                    #EXPLORATORY DATA ANALYSIS


#Univariate Analysis


plt.figure(figsize=(10,5))
sns.histplot(df['age'].dropna(), kde = True , bins = 20 , color = 'skyblue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

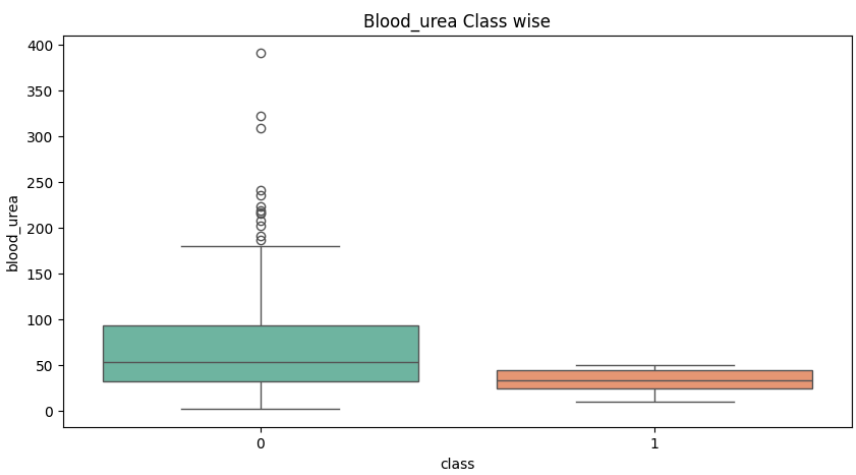



```
#we can see that Distribution of age is right skewed


#Hypertension
```

```
plt.figure(figsize=(10,5))
sns.countplot(x='hypertension', data=df,palette = 'Set2')
plt.title('Hypertension')
plt.show()
```



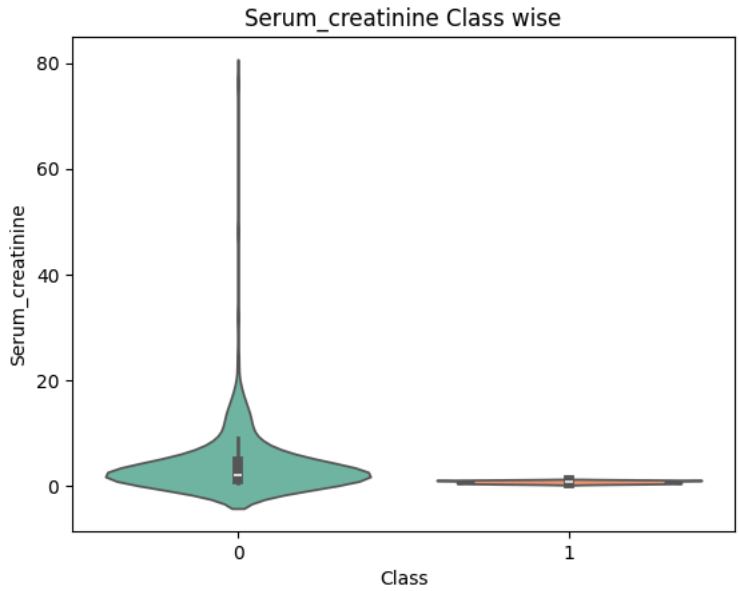

```
#Around 150 People Has hypertension


#Blood_urea Class wise


plt.figure(figsize=(10,5))
sns.boxplot(y='blood_urea', x='class', data=df, palette = 'Set2')
plt.title('Blood_urea Class wise')
plt.show()
```



```
# Class 1 Means People Don't have chronic disease
# Class 0 Means People have chronic disease
#Outlier is present in class 0
#Blooad_urea is higher in class 0 people
'''
Some of the most common chronic diseases include:
Heart disease
Cancer
Stroke
Diabetes
Arthritis
Chronic obstructive pulmonary disease (COPD)
Asthma
Mental health conditions like depression and anxiety
'''
```

```
'\nSome of the most common chronic diseases include:\nHeart disease\nCancer\nStroke
\nDiabetes\nArthritis\nChronic obstructive pulmonary disease (COPD)\nAsthma\nMental
health conditions like depression and anxiety\n'
```

```
#vIOLIN PLOT


sns.violinplot(x = 'class' , y = 'serum_creatinine' , data = df , palette = 'Set2')
plt.xlabel('Class')
plt.ylabel('Serum_creatinine')
plt.title('Serum_creatinine Class wise')
plt.show()
```

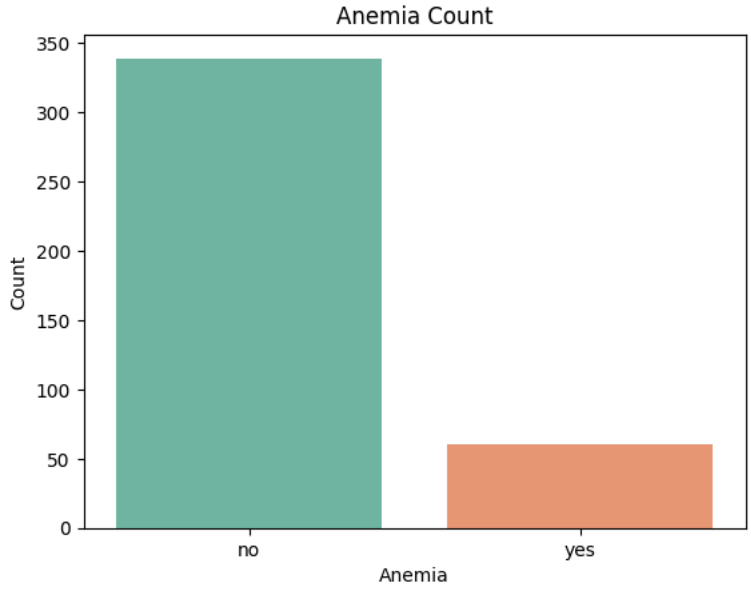



```
'''
1)Serum creatinine is a waste product in your blood that comes from the breakdown of creatine, a molecule your muscles use for energy.
Healthy kidneys filter creatinine out of your blood and eliminate it through your urine.

2) High Level of this is not a Good Sign for Kidney

3) Class 0 people has high serum creatinine with some outliers
'''
```

```
'\n1)Serum creatinine is a waste product in your blood that comes from the breakdown
of creatine, a molecule your muscles use for energy. \nHealthy kidneys filter creati
nine out of your blood and eliminate it through your urine.\n\n2) High Level of this
is not a Good Sign for Kidney\n\n3) Class 0 people has high serum creatinine with so
```

```
#Aanemia


sns.countplot(x = 'aanemia' , data = df , palette = 'Set2')
plt.xlabel('Anemia')
plt.ylabel('Count')
plt.title('Anemia Count')
```

```
Text(0.5, 1.0, 'Anemia Count')
```



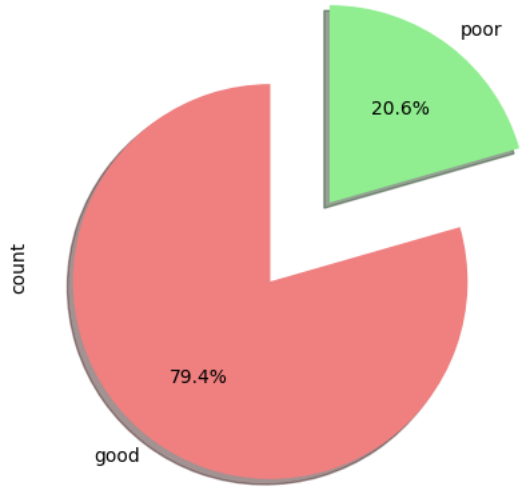

```
'''
Anemia is a blood disorder characterized by a deficiency of red blood cells (RBCs) or hemoglobin,
the protein within RBCs that carries oxygen throughout your body.
This deficiency leads to a reduction in the amount of oxygen delivered to your tissues.
There are only 60 People with Anemia
'''
```

```
'\nAnemia is a blood disorder characterized by a deficiency of red blood cells (RBC
s) or hemoglobin, \nthe protein within RBCs that carries oxygen throughout your bod
y. \nThis deficiency leads to a reduction in the amount of oxygen delivered to your
tissues. \nThere are only 60 People with Anemia\n'
```

```
#Appetite
```

```
df.appetite.value_counts().plot.pie(autopct = '%1.1f%%', colors = ['lightcoral', 'lightgreen'], explode = (0.1, 0.4), startangle = 90,
```

```
<Axes: ylabel='count'>
```

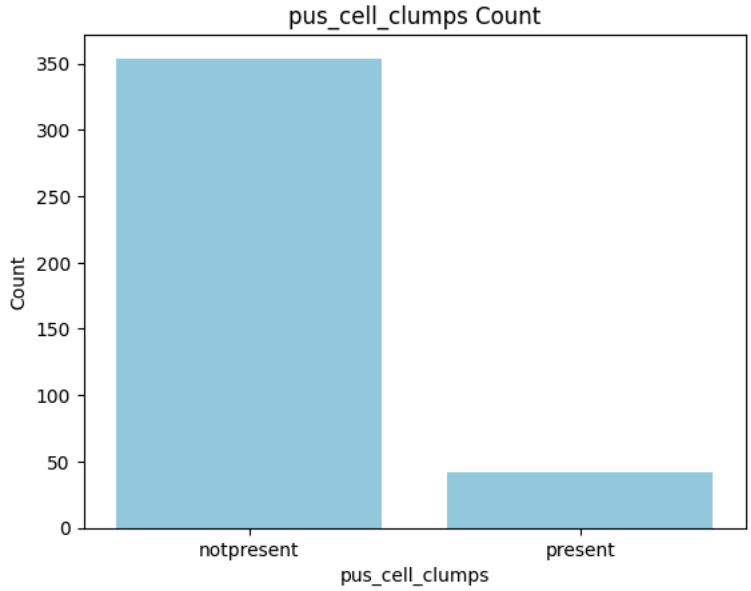



```
#79.4 Percentage have Healthy Appetite
#20.6 people have lower Appetite
```

```
# pus_cell_clumps
```

```
sns.countplot(x = 'pus_cell_clumps', data = df , color = 'skyblue')
plt.xlabel('pus_cell_clumps')
plt.ylabel('Count')
plt.title('pus_cell_clumps Count')
```

```
Text(0.5, 1.0, 'pus_cell_clumps Count')
```

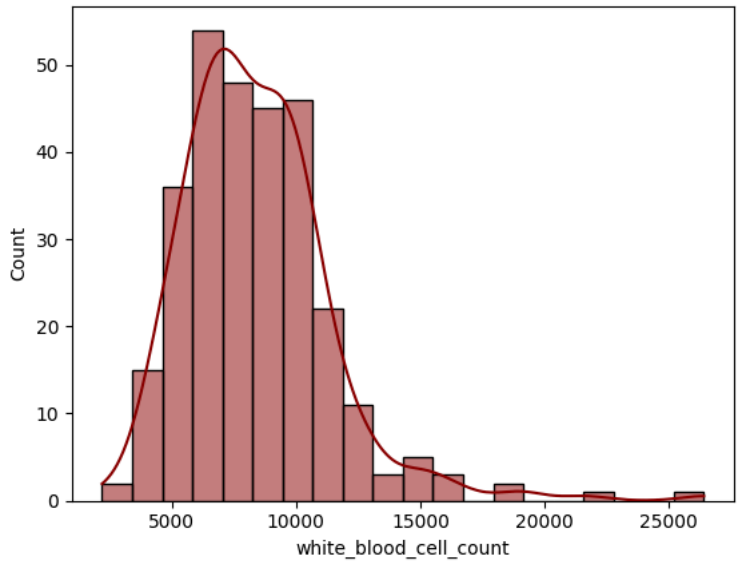



```
#pus_cell_clumps is not present in most of the people
```

```
#White-Blood-Cells
```

```
sns.histplot(df['white_blood_cell_count'].dropna(), bins = 20, kde = True, color = 'darkred')
```

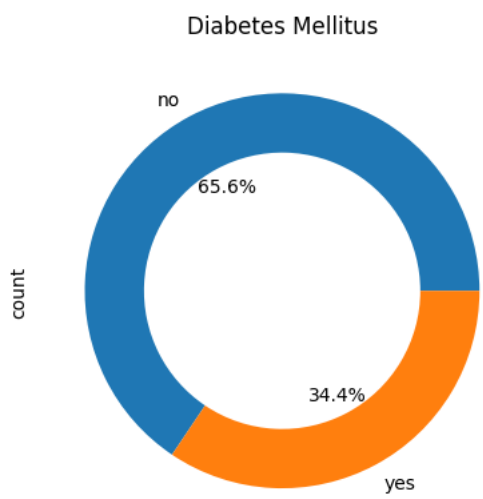
```
<Axes: xlabel='white_blood_cell_count', ylabel='Count'>
```




```
#Around 150 Peoples have the White Blood cell count between 5000 to 10000
```

```
#diabetes_mellitus
```

```
df['diabetes_mellitus'].value_counts().plot.pie(autopct = "%1.1f%%", wedgeprops = dict(width=0.3))
plt.title('Diabetes Mellitus')
plt.show()
```

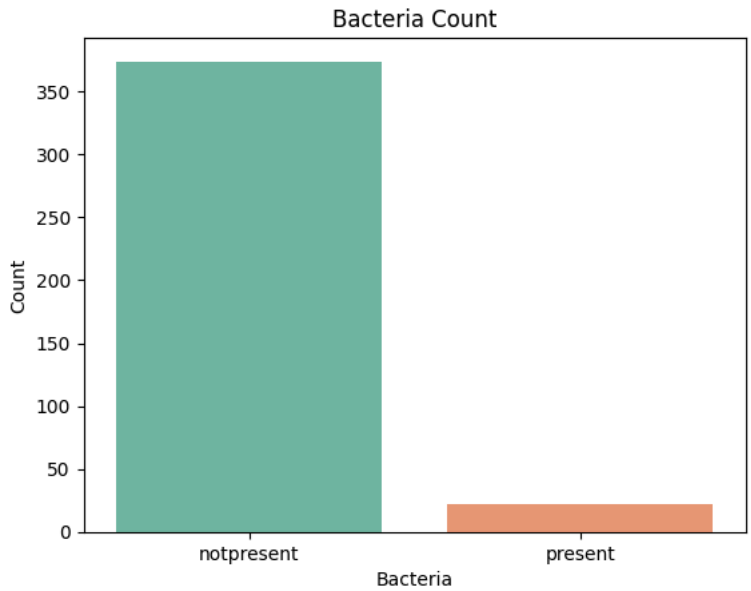



```
#65.5 People have Daibetes
```

```
#Bacteria
```

```
sns.countplot(x = 'bacteria', data=df, palette = 'Set2')
plt.xlabel('Bacteria')
plt.ylabel('Count')
plt.title('Bacteria Count')
```

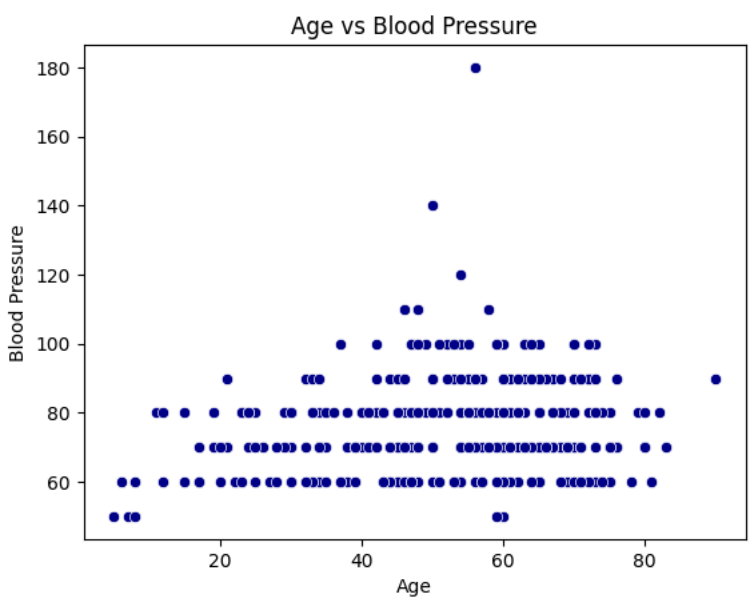
```
Text(0.5, 1.0, 'Bacteria Count')
```




```
#BIVARIATE ANALYSIS
```

```
#Age wise Blood Pressure
```

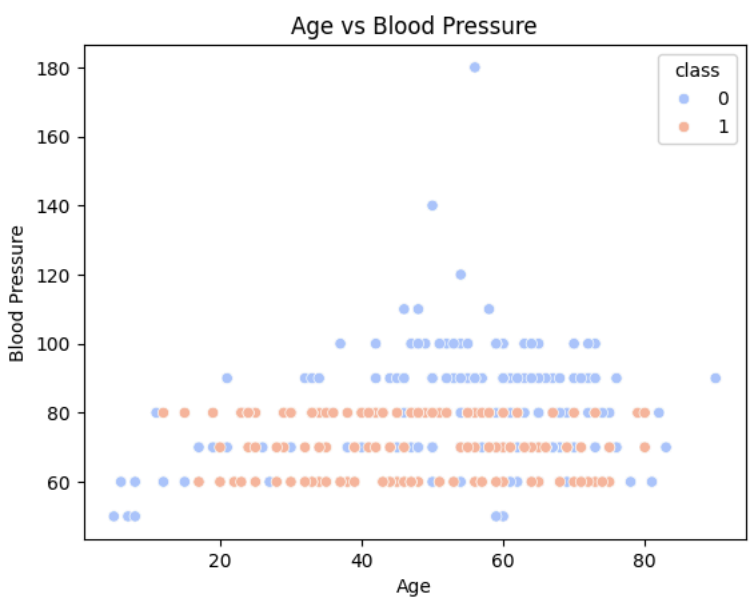
```
sns.scatterplot(x = 'age', y = 'blood_pressure', data = df, color = 'darkblue')
plt.xlabel('Age')
plt.ylabel('Blood Pressure')
plt.title('Age vs Blood Pressure')
plt.show()
```




```
#With Increase in Age Blooad Presure also increasing wit hsome outliers
```

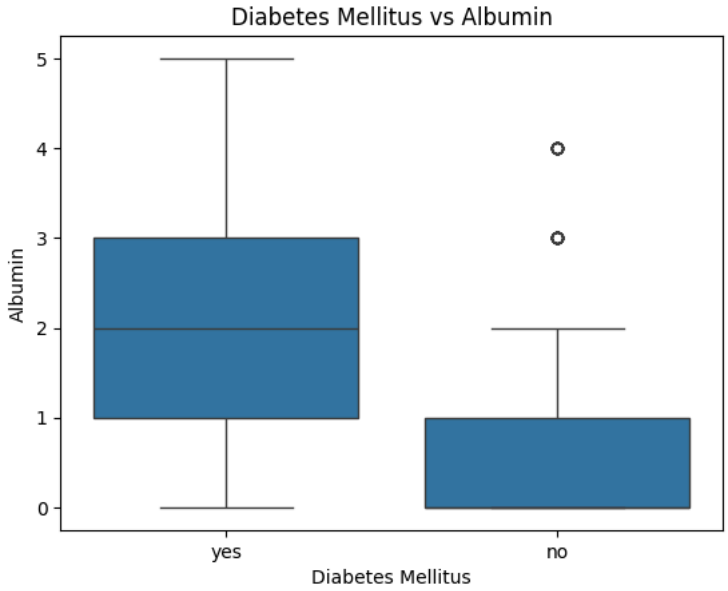
```
#Class wise Blood Pressure across different Age
```

```
sns.scatterplot(x = 'age', y = 'blood_pressure', data = df, hue = 'class', palette = 'coolwarm')
plt.xlabel('Age')
plt.ylabel('Blood Pressure')
plt.title('Age vs Blood Pressure')
plt.show()
```




```
#Class 0 People has higher blood pressure than class 1
```

```
sns.boxplot(x = 'diabetes_mellitus', y = 'albumin', data =df)
plt.xlabel('Diabetes Mellitus')
plt.ylabel('Albumin')
plt.title('Diabetes Mellitus vs Albumin')
plt.show()
```

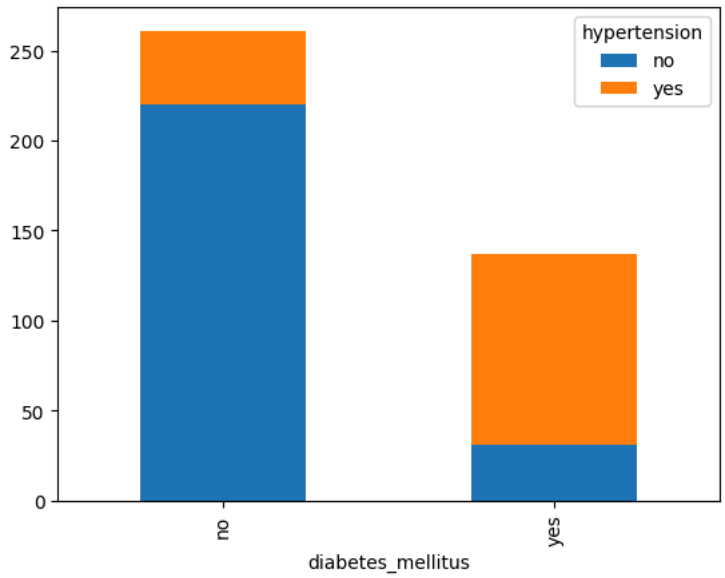



```
#People having Diabetes have more Albumin
```

```
# Diabetes Vs HyperTension
```

```
#stacked bar chart
diabetes_hypertension = pd.crosstab(df['diabetes_mellitus'], df['hypertension'])

diabetes_hypertension.plot(kind = 'bar', stacked = True)
```

```
<Axes: xlabel='diabetes_mellitus'>
```

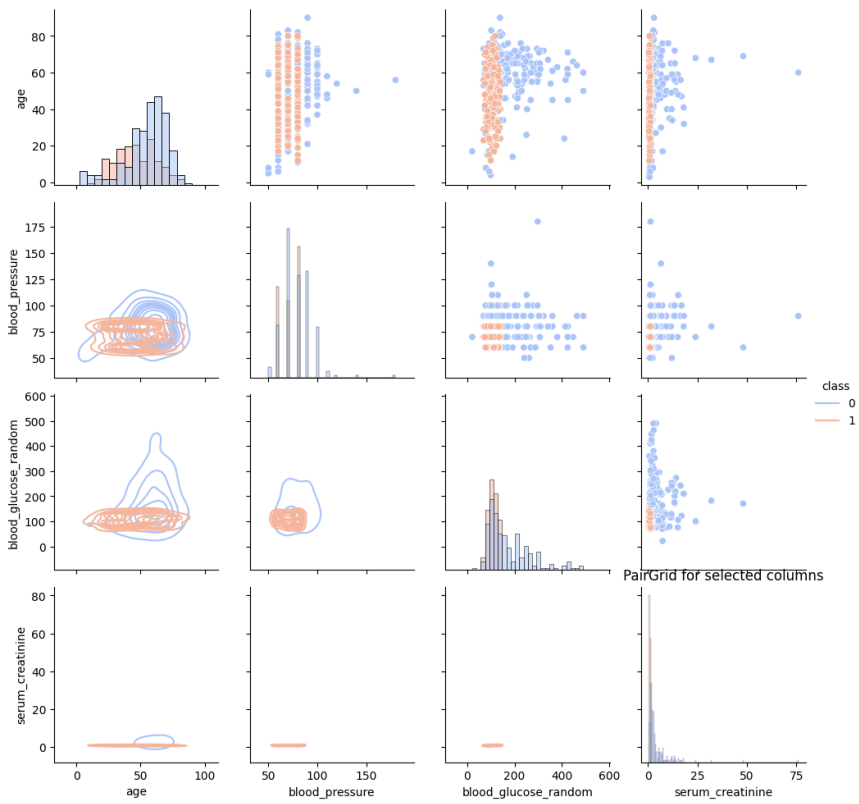



```
# The People having Diabetes has the higher chances of being Hypertension
```

```
#MULTIVARIATE ANALYSIS
```

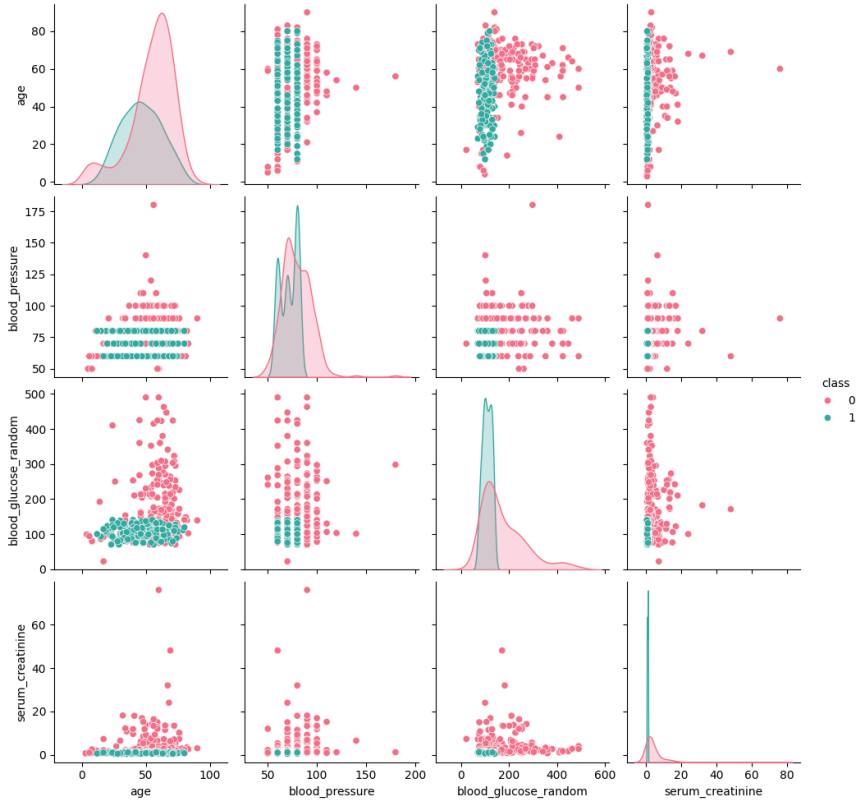
```
#Pairplot
```

```
cols = ['age', 'blood_pressure', 'blood_glucose_random', 'serum_creatinine', 'class']

g = sns.PairGrid(df[cols], hue='class', palette = 'coolwarm')
g.map_upper(sns.scatterplot)
g.map_lower(sns.kdeplot, cmap = 'Blues_d')
g.map_diag(sns.histplot)
g.add_legend()
plt.title("PairGrid for selected columns")
plt.show()
```



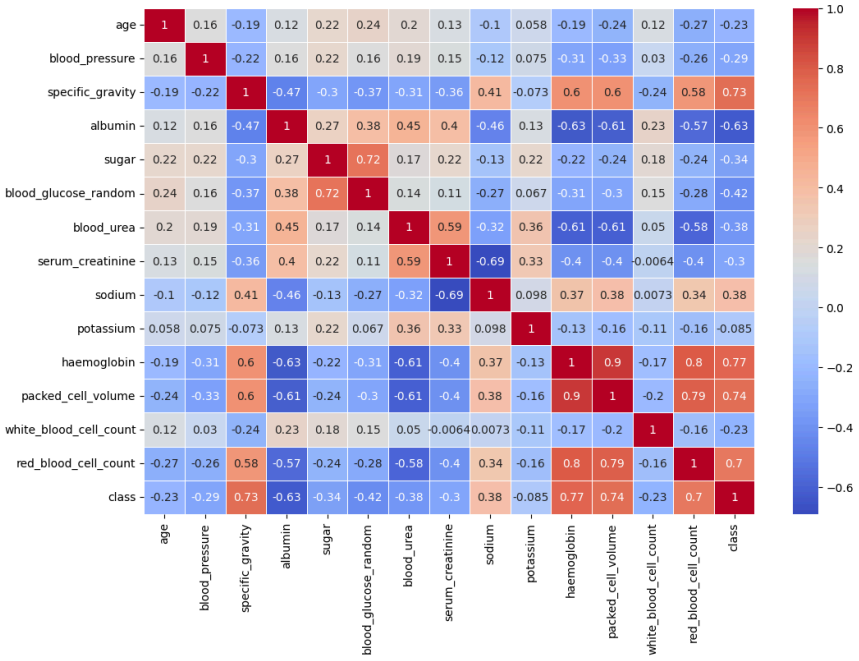

```
cols = ['age', 'blood_pressure', 'blood_glucose_random', 'serum_creatinine', 'class']
sns.pairplot(df[cols], hue = 'class', palette = 'husl')
```

```
<seaborn.axisgrid.PairGrid at 0x78ad473a95d0>
```



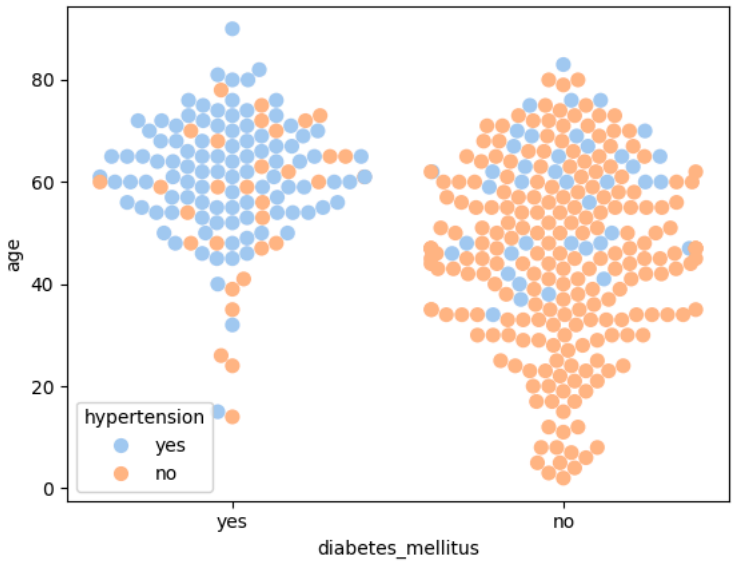

```
#Correaltion tells the relation b/w two continous Variable , it ranges from low to high i.e -1 to 1


corr = df.corr(numeric_only=True)
plt.figure(figsize = (12, 8))
sns.heatmap(corr, annot = True, cmap = 'coolwarm', linewidth = .5)
```

```
<Axes: >
```

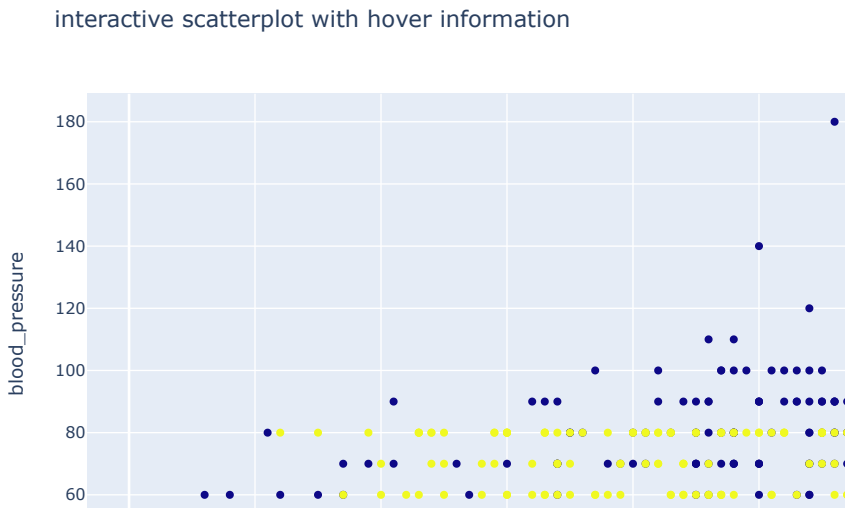



```
sns.swarmplot(x='diabetes_mellitus', y='age', hue='hypertension', data=df, palette='pastel', size=8)
```

```
<Axes: xlabel='diabetes_mellitus', ylabel='age'>
```




```
fig = px.scatter(df, x = 'age', y = 'blood_pressure', color = 'class', hover_data = ['serum_creatinine', 'haemoglobin'],title = "intera
fig.show()
```




```
df.isnull().sum()
```

```
age                         9
blood_pressure             12
specific_gravity           47
albumin                    46
sugar                      49
red_blood_cells           152
pus_cell                   65
pus_cell_clumps             4
bacteria                    4
blood_glucose_random       44
blood_urea                 19
serum_creatinine           17
sodium                     87
potassium                  88
haemoglobin                52
packed_cell_volume         71
white_blood_cell_count    106
red_blood_cell_count      131
hypertension                2
diabetes_mellitus           2
coronary_artery_disease     2
appetite                    1
peda_edema                  1
aanemia                     1
class                       0
dtype: int64
```

```
cat_cols
```

```
['red_blood_cells',
 'pus_cell',
 'pus_cell_clumps',
 'bacteria',
 'packed_cell_volume',
 'white_blood_cell_count',
 'red_blood_cell_count',
 'hypertension',
 'diabetes_mellitus',
 'coronary_artery_disease',
 'appetite',
 'peda_edema',
 'aanemia',
 'class']
```

```
num_cols
```

```
['age',
 'blood_pressure',
 'specific_gravity',
 'albumin'
```