MANOJ KUMAR - 2048015

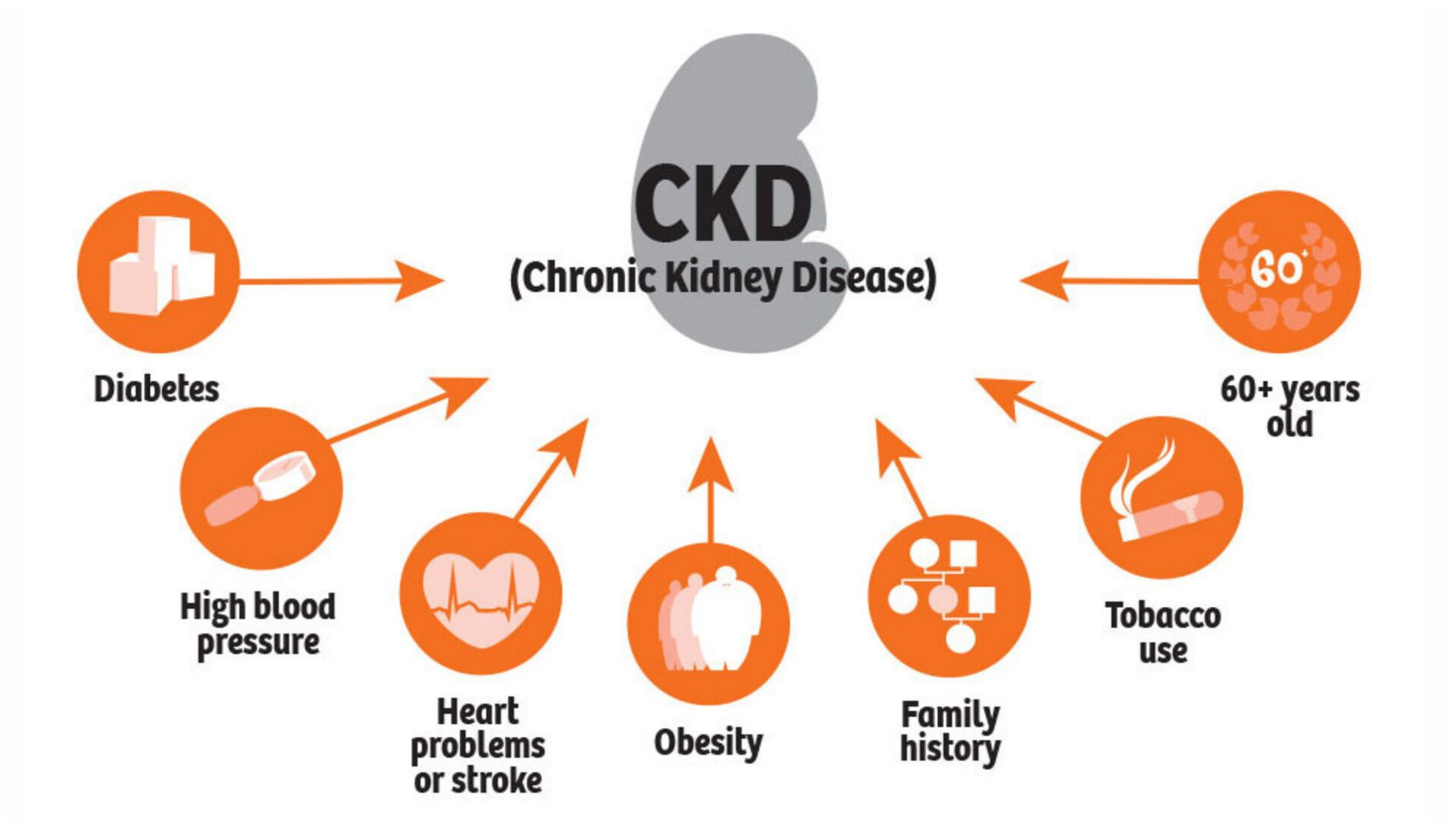
Exploratory Data Analysis of Chronic Kidney Disease

Introduction

Chronic kidney disease (CKD), also known as chronic renal disease. Chronic kidney disease involves conditions that damage your kidneys and decrease their ability to keep you healthy.

Our kidney is involved in multiple key functions.

- They help to maintain overall fluid balance and help to regulate and filter minerals from blood
- They help to filter wastes generated from medications, food and toxic substances
- They then help in creating hormones that help produce red blood cells, promote bone health, and regulate blood pressure



Overview

- The Chronic Kidney Disease Dataset consists of 24 features and one target variable.
- It is a **binary classification** problem
- The numerical features include: 'age', 'blood_pressure', 'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium', 'potassium', 'hemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count'

- The categorical features include: 'specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria', 'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite', 'pedal_edema', 'anemia', 'classification'

Content

Importing packages

In [1]:
```
import numpy as np
import pandas as pd

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

In [2]:
```
df.shape
```

Out[2]:
```
(400, 26)
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              400 non-null    int64  
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object 
 7   pc              335 non-null    object 
 8   pcc             396 non-null    object 
 9   ba              396 non-null    object 
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object 
 17  wc              295 non-null    object 
 18  rc              270 non-null    object 
 19  htn             398 non-null    object 
 20  dm              398 non-null    object 
 21  cad             398 non-null    object 
 22  appet           399 non-null    object 
 23  pe              399 non-null    object 
 24  ane             399 non-null    object 
 25  classification  400 non-null    object 
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

In [4]:
```
df.head().T
```

Out[4]:

	0	1	2	3	4
id	0	1	2	3	4
age	48	7	62	48	51
bp	80	50	80	70	80
sg	1.02	1.02	1.01	1.005	1.01
al	1	4	2	4	2
su	0	0	3	0	0
rbc	NaN	NaN	normal	normal	normal
pc	normal	normal	normal	abnormal	normal
pcc	notpresent	notpresent	notpresent	present	notpresent
ba	notpresent	notpresent	notpresent	notpresent	notpresent
bgr	121	NaN	423	117	106
bu	36	18	53	56	26
sc	1.2	0.8	1.8	3.8	1.4
sod	NaN	NaN	NaN	111	NaN
pot	NaN	NaN	NaN	2.5	NaN
hemo	15.4	11.3	9.6	11.2	11.6
pcv	44	38	31	32	35
wc	7800	6000	7500	6700	7300
rc	5.2	NaN	NaN	3.9	4.6
htn	yes	no	no	yes	no
dm	yes	no	yes	no	no
cad	no	no	no	no	no
appet	good	good	poor	poor	good
pe	no	no	no	yes	no
ane	no	no	yes	yes	no
classification	ckd	ckd	ckd	ckd	ckd

1. Features:

1. **age** - age
2. **bp** - blood pressure
3. **sg** - specific gravity
4. **al** - albumin
5. **su** - sugar
6. **rbc** - red blood cells
7. **pc** - pus cell
8. **pcc** - pus cell clumps
9. **ba** - bacteria
10. **bgr** - blood glucose random
11. **bu** - blood urea
12. **sc** - serum creatinine
13. **sod** - sodium
14. **pot** - potassium
15. **hemo** - haemoglobin
16. **pcv** - packed cell volume
17. **wc** - white blood cell count
18. **rc** - red blood cell count
19. **htn** - hypertension
20. **dm** - diabetes mellitus
21. **cad** - coronary artery disease
22. **appet** - appetite
23. **pe** - pedal edema
24. **ane** - anemia
25. **classification** - class

2. Feature description

1. Age(numerical) --> age in years

2. Blood Pressure(numerical) bp in mm/Hg
3. Specific Gravity(nominal) sg - (1.005,1.010,1.015,1.020,1.025)
4. Albumin(nominal)al - (0,1,2,3,4,5)
5. Sugar(nominal) su - (0,1,2,3,4,5)
6. Red Blood Cells(nominal) rbc - (normal,abnormal)
7. Pus Cell (nominal)pc - (normal,abnormal)
8. Pus Cell clumps(nominal)pcc - (present,notpresent)
9. Bacteria(nominal) ba - (present,notpresent)
10. Blood Glucose Random(numerical) bgr in mgs/dl
11. Blood Urea(numerical) bu in mgs/dl
12. Serum Creatinine(numerical) sc in mgs/dl
13. Sodium(numerical) sod in mEq/L
14. Potassium(numerical) pot in mEq/L
15. Haemoglobin(numerical) hemo in gms
16. Packed Cell Volume(numerical)
17. White Blood Cell Count(numerical) wc in cells/cumm
18. Red Blood Cell Count(numerical) rc in millions/cmm
19. Hypertension(nominal) htn - (yes,no)
20. Diabetes Mellitus(nominal) dm - (yes,no)
21. Coronary Artery Disease(nominal) cad - (yes,no)
22. Appetite(nominal) ppet - (good,poor)
23. Pedal Edema(nominal) pe - (yes,no)
24. Anemia(nominal)ane - (yes,no)
25. Class (nominal) class - (ckd,notckd)

In [7]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

3. Rename the columns to have meaningful names & checking data types

In [17]:
```
cols_names={"bp":"blood_pressure",
          "sg":"specific_gravity",
          "al":"albumin",
          "su":"sugar",
          "rbc":"red_blood_cells",
          "pc":"pus_cell",
          "pcc":"pus_cell_clumps",
          "ba":"bacteria",
          "bgr":"blood_glucose_random",
          "bu":"blood_urea",
          "sc":"serum_creatinine",
          "sod":"sodium",
          "pot":"potassium",
          "hemo":"haemoglobin",
          "pcv":"packed_cell_volume",
          "wc":"white_blood_cell_count",
          "rc":"red_blood_cell_count",
          "htn":"hypertension",
          "dm":"diabetes_mellitus",
          "cad":"coronary_artery_disease",
          "appet":"appetite",
          "pe":"pedal_edema",
          "ane":"anemia"}

df.rename(columns=cols_names, inplace=True)
```

In [22]:
```
print(f"Totally, {df.shape[1]} columns and {df.shape[0]} Rows")
```

```
Totally, 26 columns and 400 Rows
```

```
In [23]: df.head().T
```

Out[23]:

	0	1	2	3	4
id	0	1	2	3	4
age	48	7	62	48	51
blood_pressure	80	50	80	70	80
specific_gravity	1.02	1.02	1.01	1.005	1.01
albumin	1	4	2	4	2
sugar	0	0	3	0	0
red_blood_cells	NaN	NaN	normal	normal	normal
pus_cell	normal	normal	normal	abnormal	normal
pus_cell_clumps	notpresent	notpresent	notpresent	present	notpresent
bacteria	notpresent	notpresent	notpresent	notpresent	notpresent
blood_glucose_random	121	NaN	423	117	106
blood_urea	36	18	53	56	26
serum_creatinine	1.2	0.8	1.8	3.8	1.4
sodium	NaN	NaN	NaN	111	NaN
potassium	NaN	NaN	NaN	2.5	NaN
haemoglobin	15.4	11.3	9.6	11.2	11.6
packed_cell_volume	44	38	31	32	35
white_blood_cell_count	7800	6000	7500	6700	7300
red_blood_cell_count	5.2	NaN	NaN	3.9	4.6
hypertension	yes	no	no	yes	no
diabetes_mellitus	yes	no	yes	no	no
coronary_artery_disease	no	no	no	no	no
appetite	good	good	poor	poor	good
pedal_edema	no	no	no	yes	no
anemia	no	no	yes	yes	no
classification	ckd	ckd	ckd	ckd	ckd

```
In [25]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       400 non-null    int64  
 1   age                      391 non-null    float64
 2   blood_pressure           388 non-null    float64
 3   specific_gravity         353 non-null    float64
 4   albumin                  354 non-null    float64
 5   sugar                    351 non-null    float64
 6   red_blood_cells          248 non-null    object 
 7   pus_cell                 335 non-null    object 
 8   pus_cell_clumps          396 non-null    object 
 9   bacteria                 396 non-null    object 
 10  blood_glucose_random     356 non-null    float64
 11  blood_urea               381 non-null    float64
 12  serum_creatinine         383 non-null    float64
 13  sodium                   313 non-null    float64
 14  potassium                312 non-null    float64
 15  haemoglobin              348 non-null    float64
 16  packed_cell_volume       330 non-null    object 
 17  white_blood_cell_count   295 non-null    object 
 18  red_blood_cell_count     270 non-null    object 
 19  hypertension             398 non-null    object 
 20  diabetes_mellitus        398 non-null    object 
 21  coronary_artery_disease  398 non-null    object 
 22  appetite                 399 non-null    object 
 23  pedal_edema              399 non-null    object 
 24  anemia                   399 non-null    object 
 25  classification           400 non-null    object 
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

red_blood_cell_count, packed_cell_volume and white_blood_cell_count are object type.
We need to change to numerical dtype.

In [32]:
```
df['red_blood_cell_count'] = pd.to_numeric(df['red_blood_cell_count'], errors='coerce')
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')
```

Id column is seems to be an unique identifier for each row so we are dropping that it won't help us to find any insights from the data.

In [33]:
```
# Dropping the id column

df.drop(["id"],axis=1,inplace=True)
```

4. Checking missing values

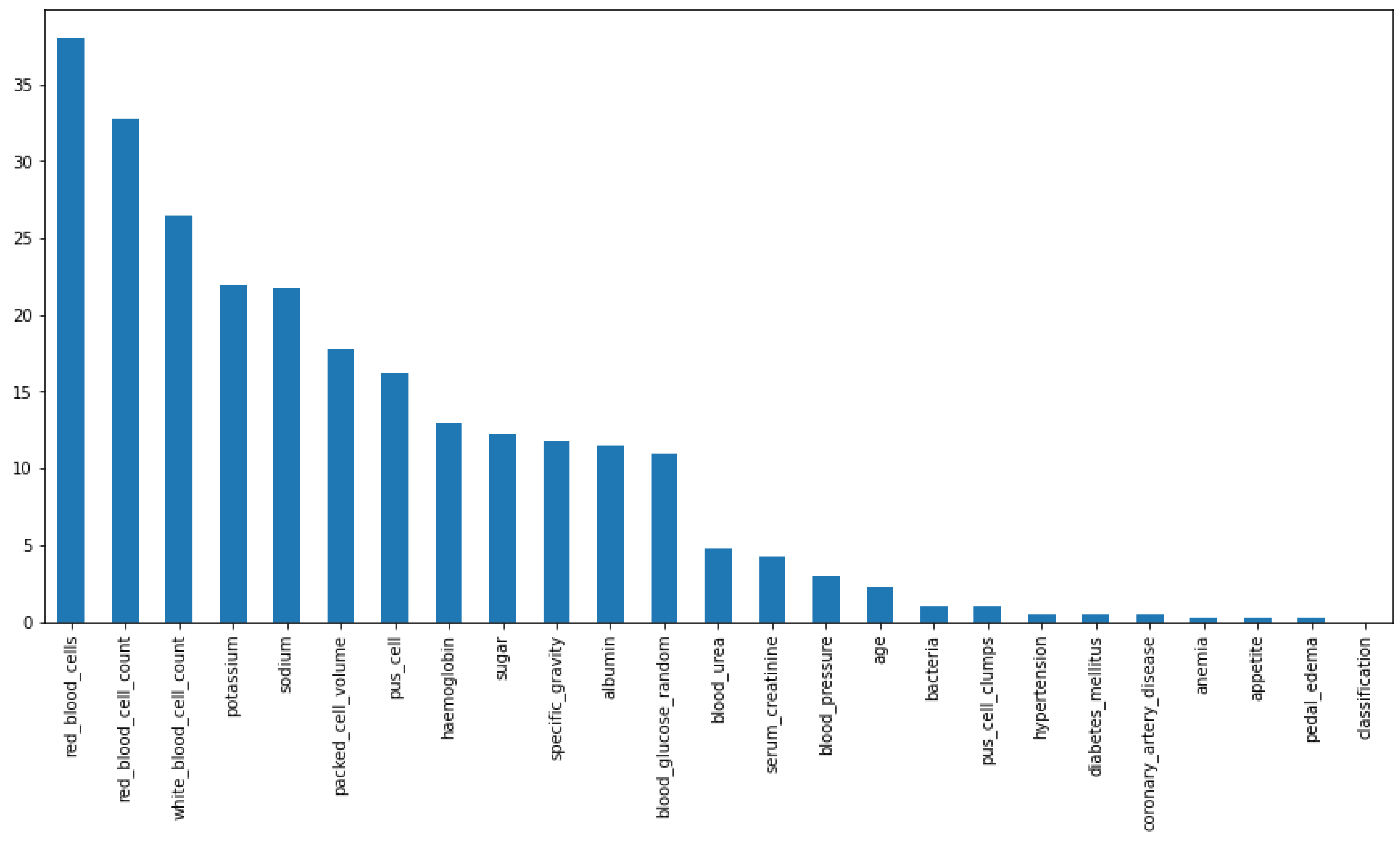
In [36]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[36]:
```
red_blood_cells            152
red_blood_cell_count       131
white_blood_cell_count     106
potassium                   88
sodium                      87
packed_cell_volume          71
pus_cell                    65
haemoglobin                 52
sugar                       49
specific_gravity            47
albumin                     46
blood_glucose_random        44
blood_urea                  19
serum_creatinine            17
blood_pressure              12
age                          9
bacteria                     4
pus_cell_clumps              4
hypertension                 2
diabetes_mellitus            2
coronary_artery_disease      2
anemia                       1
appetite                     1
pedal_edema                  1
classification               0
dtype: int64
```

Plotting missing values percentage for all the features

In [38]:
```
((df.isnull().sum()/df.shape[0])*100).sort_values(ascending=False).plot(kind='bar', figsize=(15,7))
```

Out[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb85505d6d0>
```



5. Checking for Unique values and categorical features

```
In [39]: for i in df.columns:
             print(f'{i} \n\t\t : \t {df[i].nunique()} values')
```

```
age
                 :       76 values
blood_pressure
                 :       10 values
specific_gravity
                 :       5 values
albumin
                 :       6 values
sugar
                 :       6 values
red_blood_cells
                 :       2 values
pus_cell
                 :       2 values
pus_cell_clumps
                 :       2 values
bacteria
                 :       2 values
blood_glucose_random
                 :       146 values
blood_urea
                 :       118 values
serum_creatinine
                 :       84 values
sodium
                 :       34 values
potassium
                 :       40 values
haemoglobin
                 :       115 values
packed_cell_volume
                 :       42 values
white_blood_cell_count
                 :       89 values
red_blood_cell_count
                 :       45 values
hypertension
                 :       2 values
diabetes_mellitus
                 :       5 values
coronary_artery_disease
                 :       3 values
appetite
                 :       2 values
pedal_edema
                 :       2 values
anemia
                 :       2 values
classification
                 :       3 values
```

```
In [48]: numerical_features = []
         categorical_features = []

         for i in df.columns:
             if df[i].nunique()>7:
                 numerical_features.append(i)
             else:
                 categorical_features.append(i)
```

Numerical features

```
In [49]: # Numerical features:

         print(numerical_features)
```

```
['age', 'blood_pressure', 'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium', 'potassium', 'h
aemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count']
```

Categorical features

```
In [50]: # Categorical features:

         print(categorical_features)
```

```
['specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria', 'hyper
tension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite', 'pedal_edema', 'anemia', 'classificatio
n']
```

In [51]:
```python
# checking for unique values in categorical features:

for feats in categorical_features:
    print(f'{feats} has {df[feats].unique()} categories.\n')
```

```
specific_gravity has [1.02   1.01   1.005 1.015    nan 1.025] categories.

albumin has [ 1.   4.   2.   3.   0. nan   5.] categories.

sugar has [ 0.   3.   4.   1. nan   2.   5.] categories.

red_blood_cells has [nan 'normal' 'abnormal'] categories.

pus_cell has ['normal' 'abnormal' nan] categories.

pus_cell_clumps has ['notpresent' 'present' nan] categories.

bacteria has ['notpresent' 'present' nan] categories.

hypertension has ['yes' 'no' nan] categories.

diabetes_mellitus has ['yes' 'no' ' yes' '\tno' '\tyes' nan] categories.

coronary_artery_disease has ['no' 'yes' '\tno' nan] categories.

appetite has ['good' 'poor' nan] categories.

pedal_edema has ['no' 'yes' nan] categories.

anemia has ['no' 'yes' nan] categories.

classification has ['ckd' 'ckd\t' 'notckd'] categories.
```

Based on the above result, we need to correct 2 features and the target variable which contain certain discrepancy in some values.

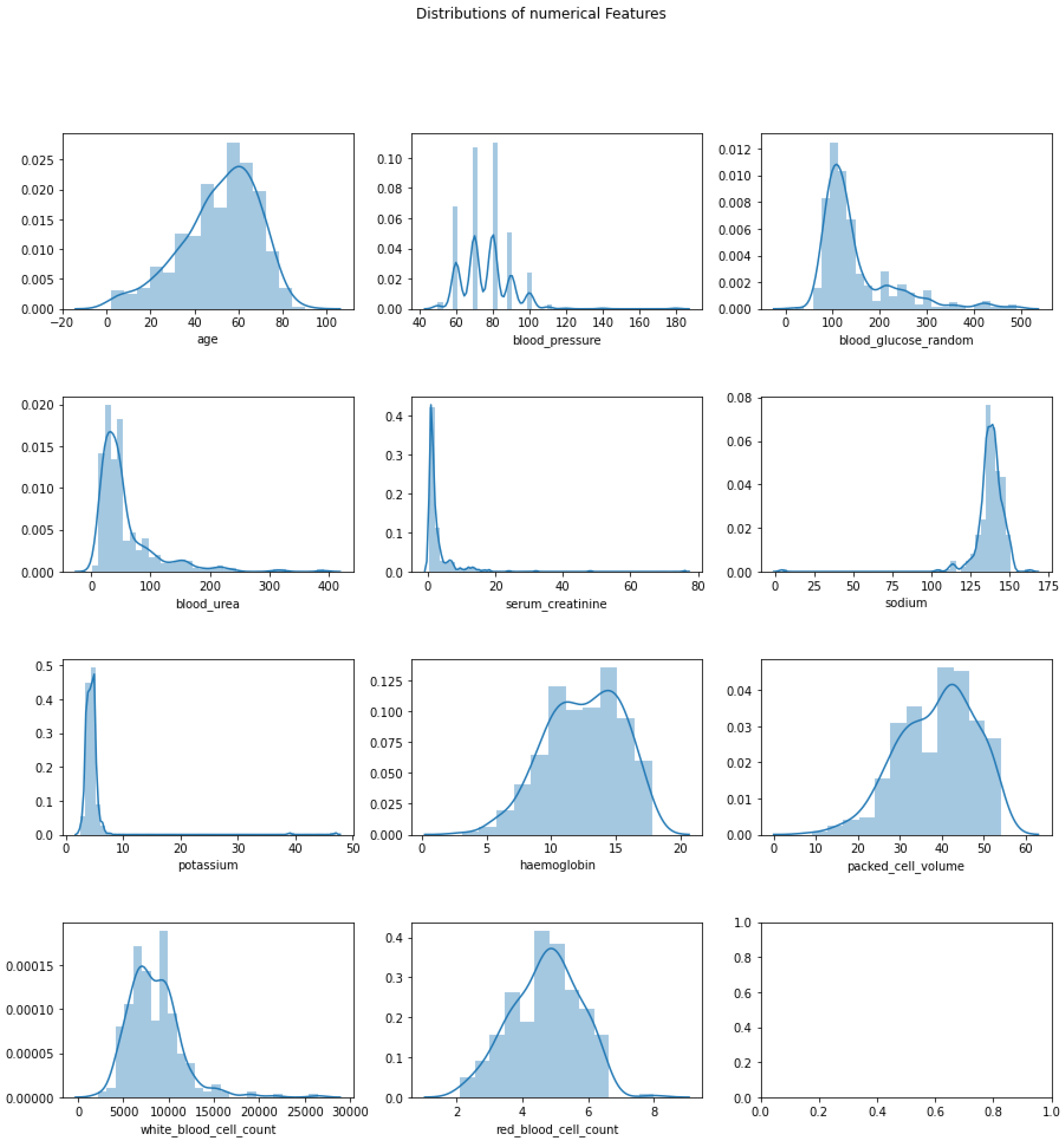
In [54]:
```python
# Replace incorrect values

df['diabetes_mellitus'] = df['diabetes_mellitus'].replace(to_replace = {'\tno':'no','\tyes':'yes',' yes':'yes'}
df['coronary_artery_disease'] = df['coronary_artery_disease'].replace(to_replace = '\tno', value='no')
df['classification'] = df['classification'].replace(to_replace = 'ckd\t', value = 'ckd')
```

In [55]:
```python
for feats in categorical_features:
    print(f'{feats} has {df[feats].unique()} categories.\n')
```

```
specific_gravity has [1.02   1.01   1.005 1.015    nan 1.025] categories.

albumin has [ 1.   4.   2.   3.   0. nan   5.] categories.

sugar has [ 0.   3.   4.   1. nan   2.   5.] categories.

red_blood_cells has [nan 'normal' 'abnormal'] categories.

pus_cell has ['normal' 'abnormal' nan] categories.

pus_cell_clumps has ['notpresent' 'present' nan] categories.

bacteria has ['notpresent' 'present' nan] categories.

hypertension has ['yes' 'no' nan] categories.

diabetes_mellitus has ['yes' 'no' nan] categories.

coronary_artery_disease has ['no' 'yes' nan] categories.

appetite has ['good' 'poor' nan] categories.

pedal_edema has ['no' 'yes' nan] categories.

anemia has ['no' 'yes' nan] categories.

classification has ['ckd' 'notckd'] categories.
```

6. Checking features distribution

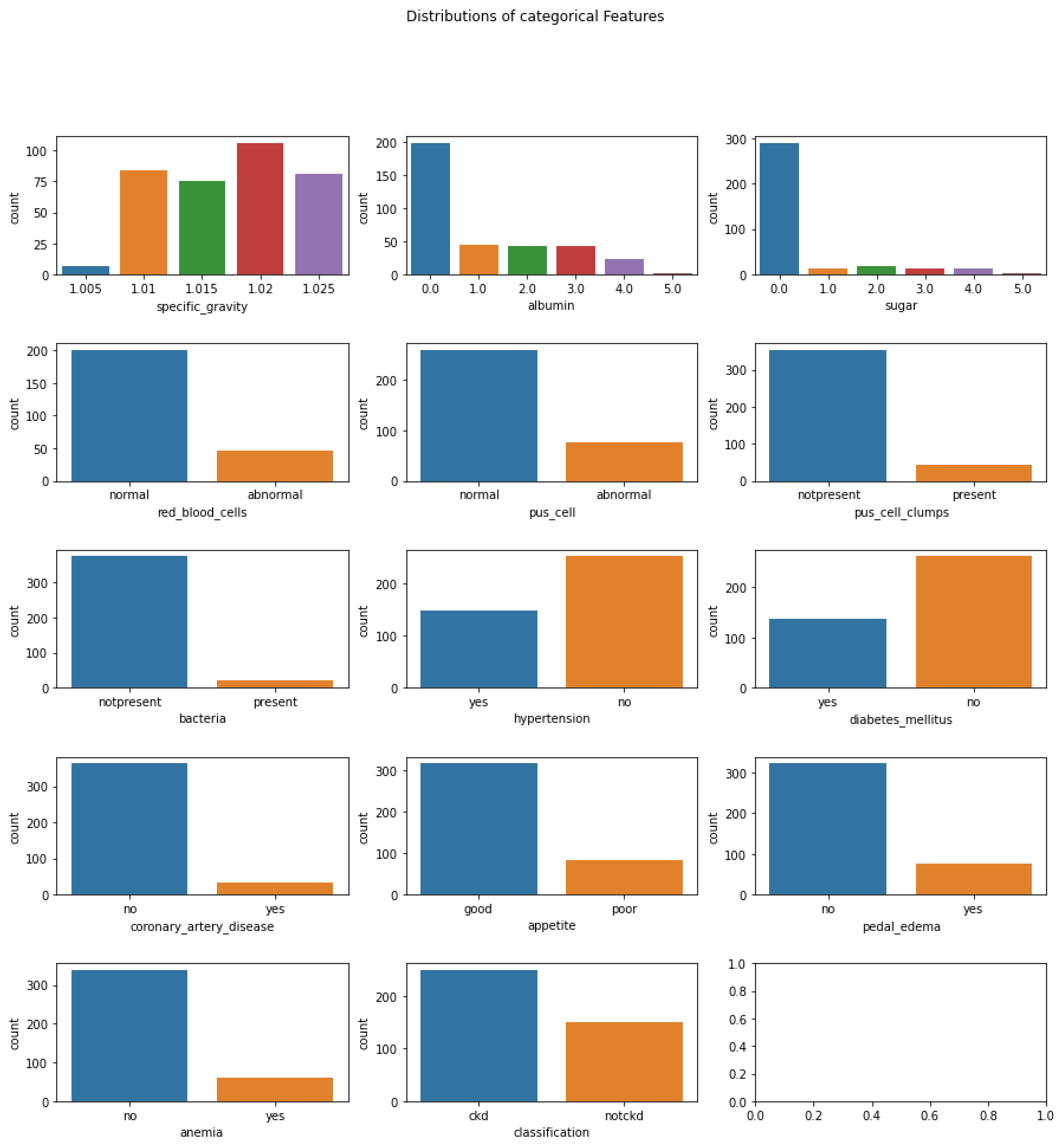
In [80]:
```
# Checking distribution of the numerical features:

fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(15,15))
fig.subplots_adjust(hspace=0.5)
fig.suptitle('Distributions of numerical Features')


for ax, feats in zip(axes.flatten(), numerical_features):
    sns.distplot(a=df[feats], ax=ax)
```



Observations:

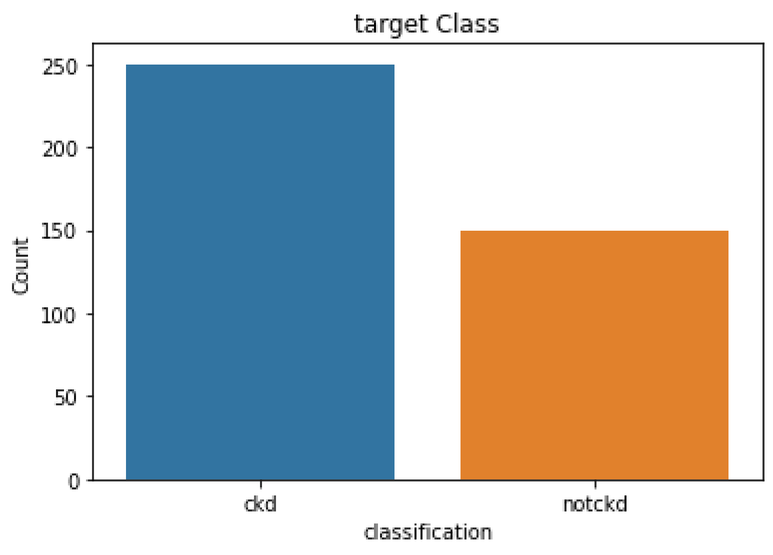
1. age looks a bit left skewed
2. Blood gluscose random is right skewed
3. Blood Urea is also a bit right skewed
4. Rest of the features are lightly skewed.

```
In [84]: # Checking the label distribution for categorical data:

         fig, axes = plt.subplots(nrows=5, ncols=3, figsize=(15,15))
         fig.subplots_adjust(hspace=0.5)
         fig.suptitle('Distributions of categorical Features')


         for ax, feats in zip(axes.flatten(), categorical_features):
             sns.countplot(df[feats], ax=ax)
```



A few features have imbalanced categories.
Hence, Stratified folds will be necessary while cross validation.

In [106]:
```
sns.countplot(x='classification',data=df)
plt.xlabel("classification")
plt.ylabel("Count")
plt.title("target Class")
plt.show()

print('Percent of chronic kidney disease sample \t: ',
      round(len(df[df['classification']=='ckd'])/len(df['classification'])*100, 2), "%")
print('Percent of not a chronic kidney disease sample \t: ',
      round(len(df[df['classification']=='notckd'])/len(df['classification'])*100, 2), "%")
```

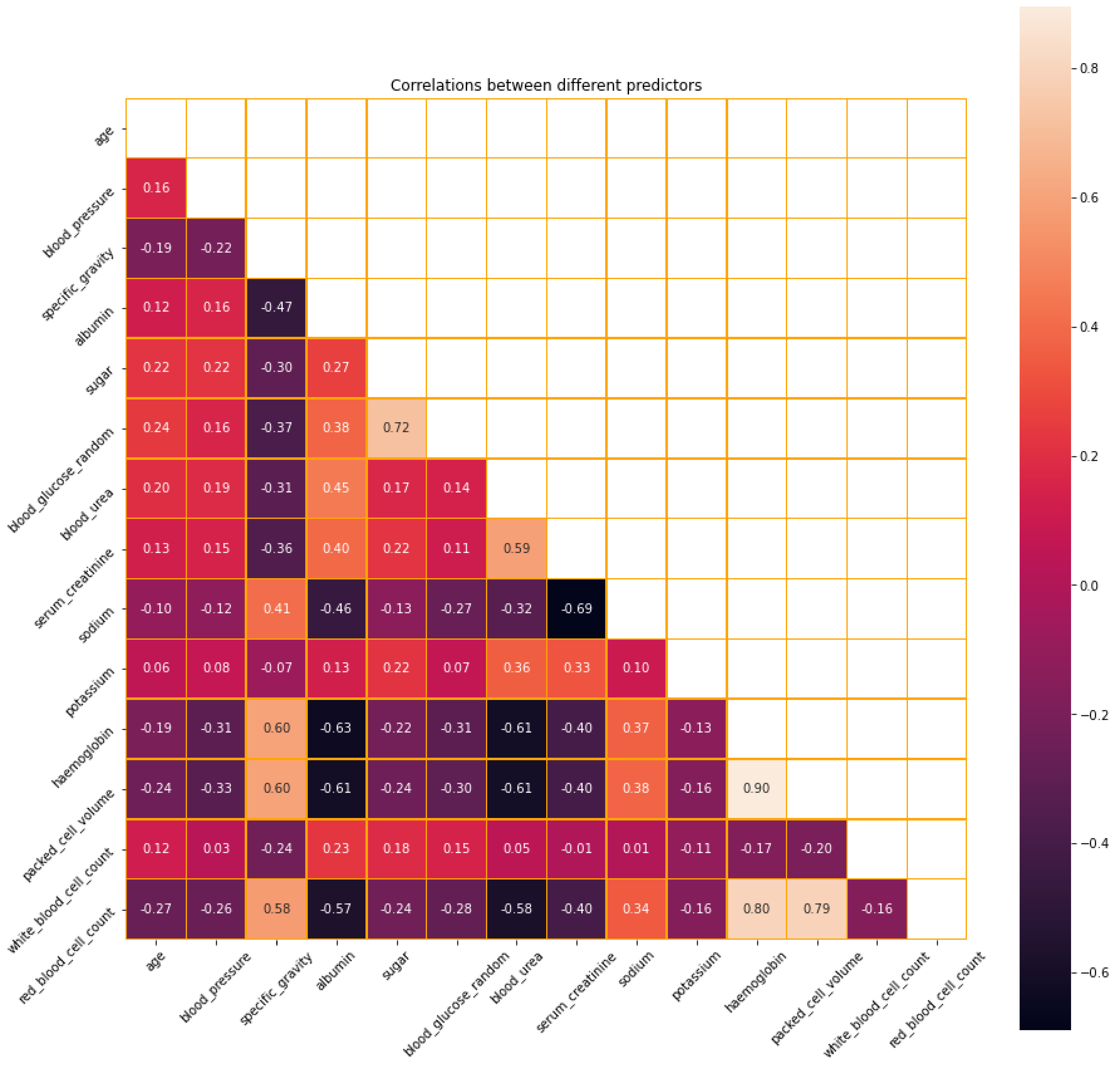


```
Percent of chronic kidney disease sample        :  62.5 %
Percent of not a chronic kidney disease sample  :  37.5 %
```

Evidently, the classes are not much imbalanced.

7. Correlations

```
In [126]: corr_df = df.corr()

          f,ax=plt.subplots(figsize=(15,15))
          mask = np.zeros_like(corr_df)
          mask[np.triu_indices_from(mask)] = True

          sns.heatmap(corr_df,annot=True,fmt=".2f",ax=ax,linewidths=0.5,linecolor="orange", mask = mask, square=True)
          plt.xticks(rotation=45)
          plt.yticks(rotation=45)
          plt.title('Correlations between different predictors')
          plt.show()
```



Positive Correlation:

1. Specific gravity -> Red blood cell count, Packed cell volume and Hemoglobin
2. Sugar -> Blood glucose random
3. Blood Urea -> Serum creatinine
4. Hemoglobin -> Red Blood cell count <- packed cell volume

Negative Correlation:

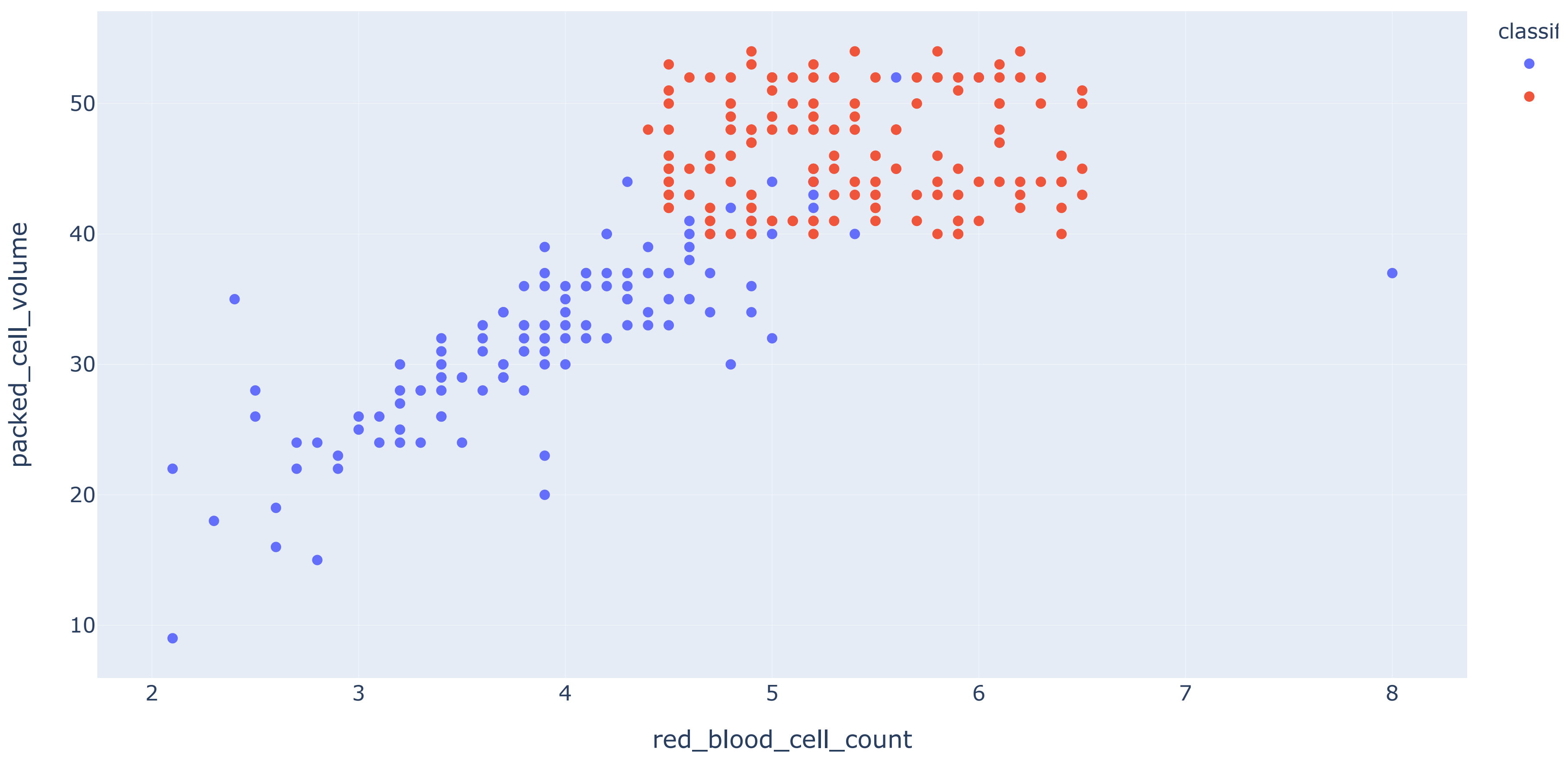
1. Albumin, Blood urea -> Red blood cell count, packed cell volume, Hemoglobin
2. Serum creatinine -> Sodium

7.1 Let's check for Positive correlation and its impact on classes

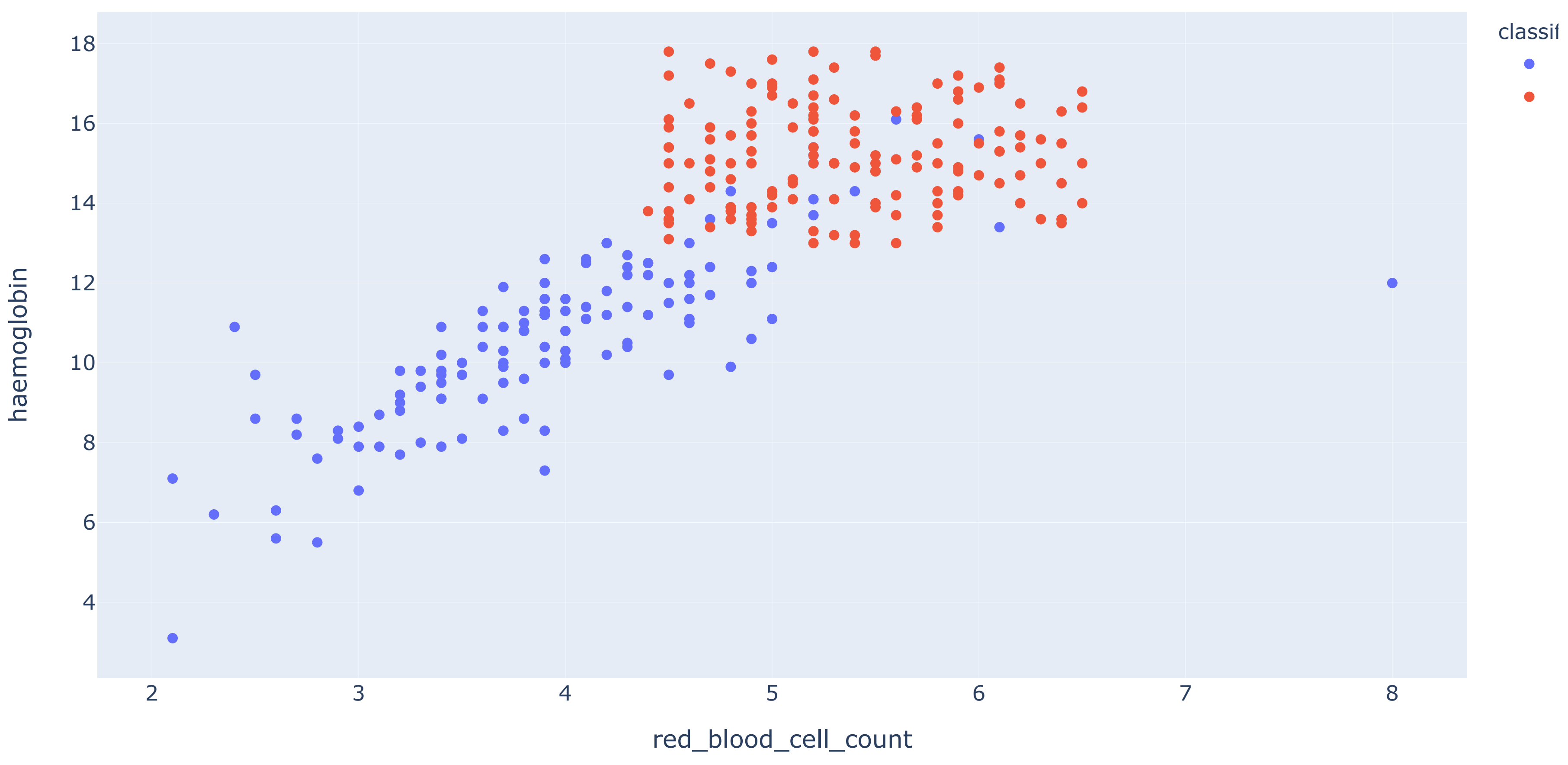
In [146]:
```
import plotly.express as px
```

In [176]:
```
# Defining violin and scatter plot functions
def violin(col):
    fig = px.violin(df,
                    y=col,
                    x="classification",
                    color="classification",
                    box=True, points="all", hover_data=df.columns)
    return fig.show()

def scatters(col1,col2):
    fig = px.scatter(df,
                     x=col1,
                     y=col2,
                     color="classification")
    fig.show()
```

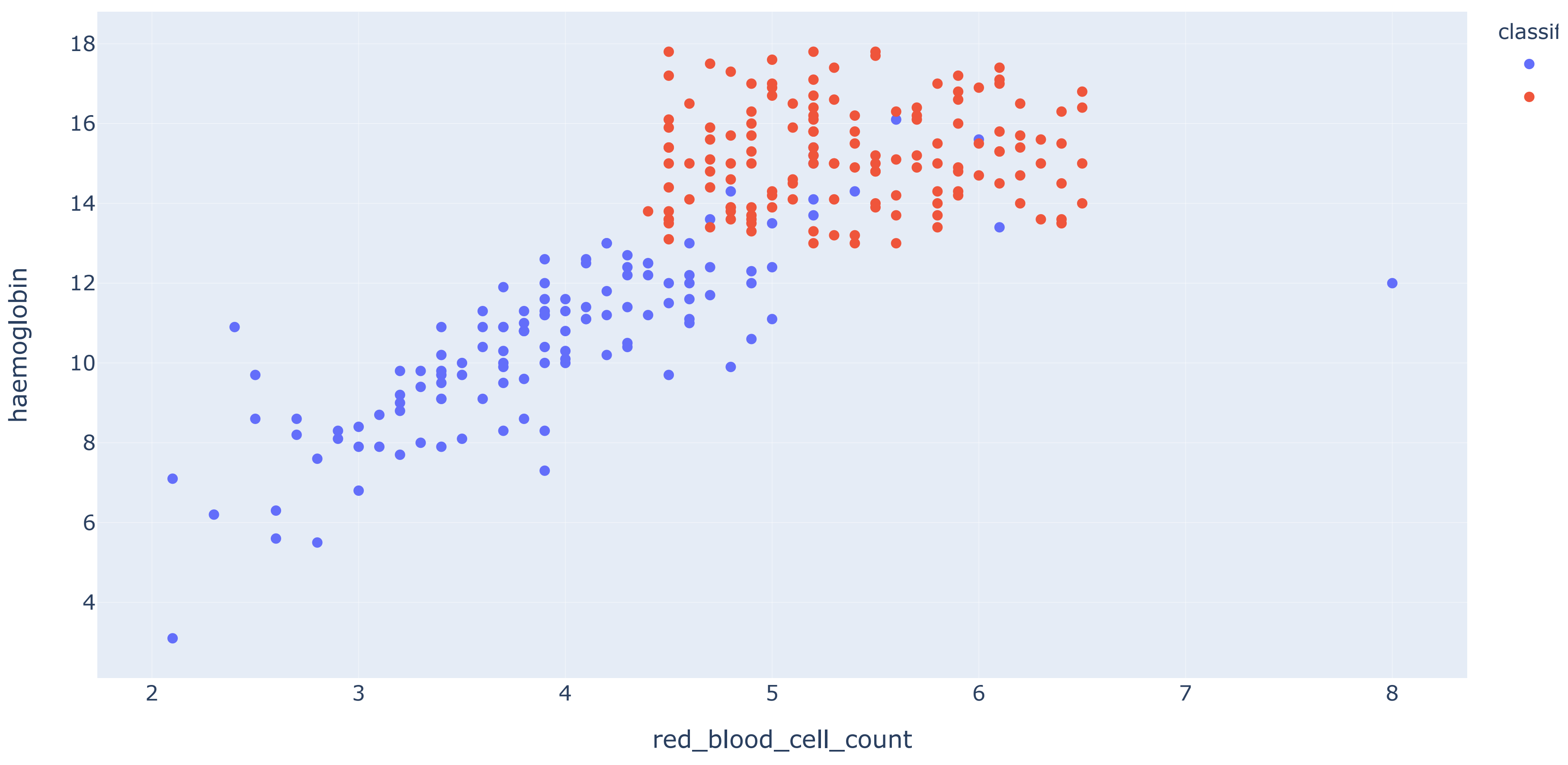
In [179]:
```
scatters('red_blood_cell_count', 'packed_cell_volume')
```



```
In [186]: scatters('red_blood_cell_count', 'haemoglobin')
```



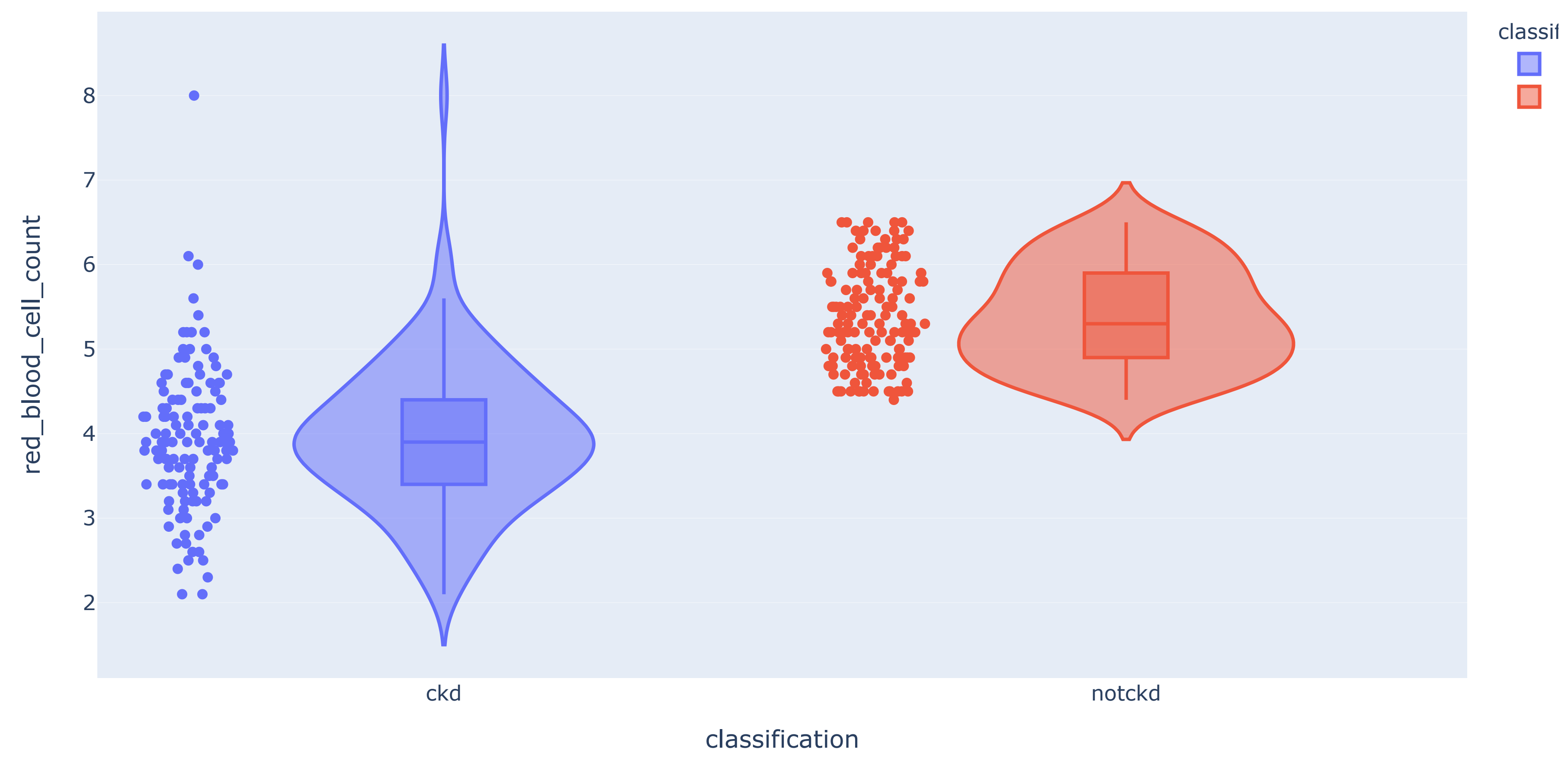
```
In [189]: scatters('red_blood_cell_count', 'haemoglobin')
```



Observations:

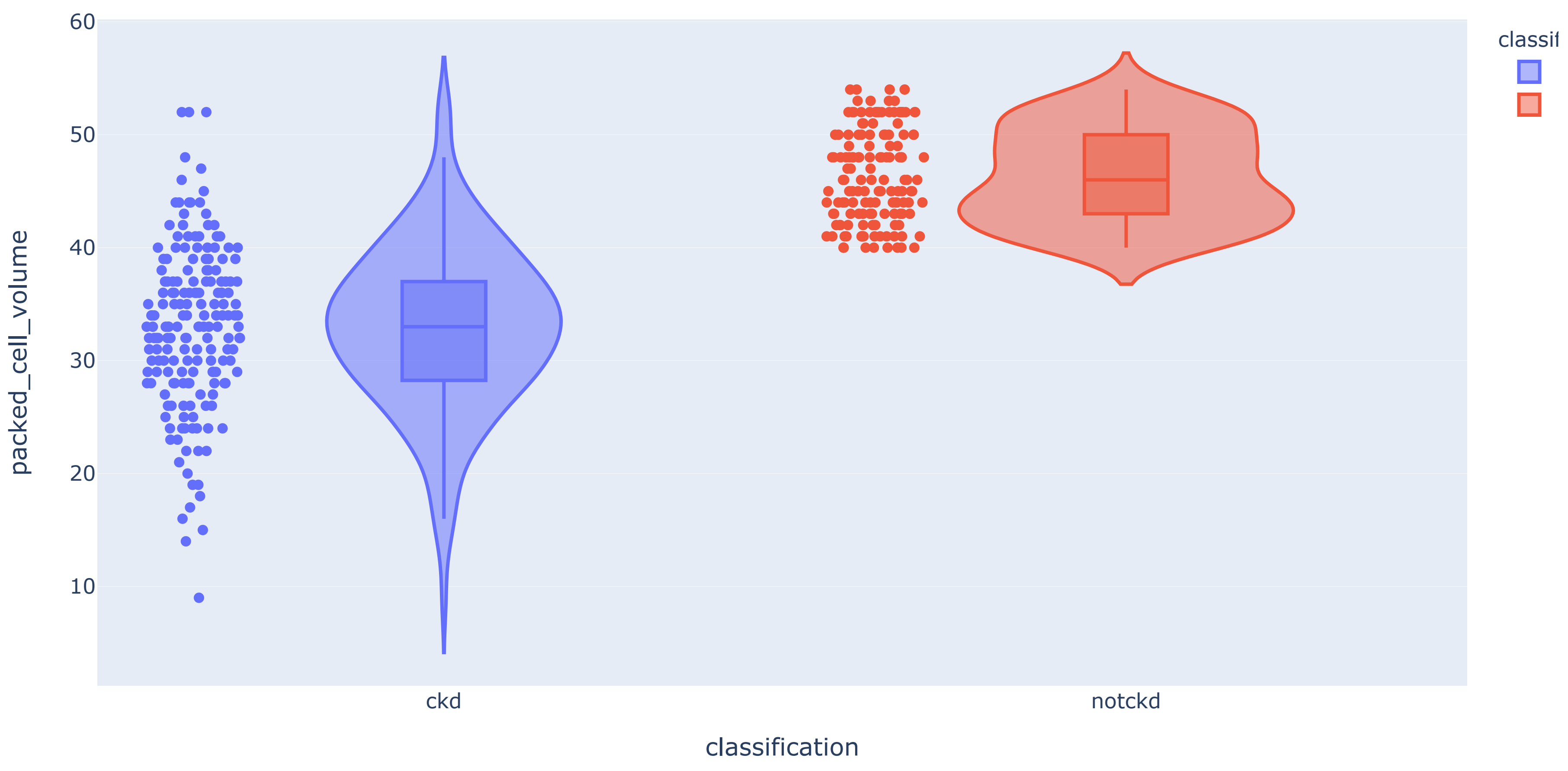
1. RBC count range ~2 to <4.5 and Hemoglobin between 3 to <13 are mostly classified as positive for chronic kidney disease(i.e ckd).
2. RBC count range >4.5 to ~6.1 and Hemoglobin between >13 to 17.8 are classified as negative for chronic kidney disease(i.e nockd).
3. Hemoglobin > 13, mostly classified as not ckd

```
In [191]: violin('red_blood_cell_count')
```

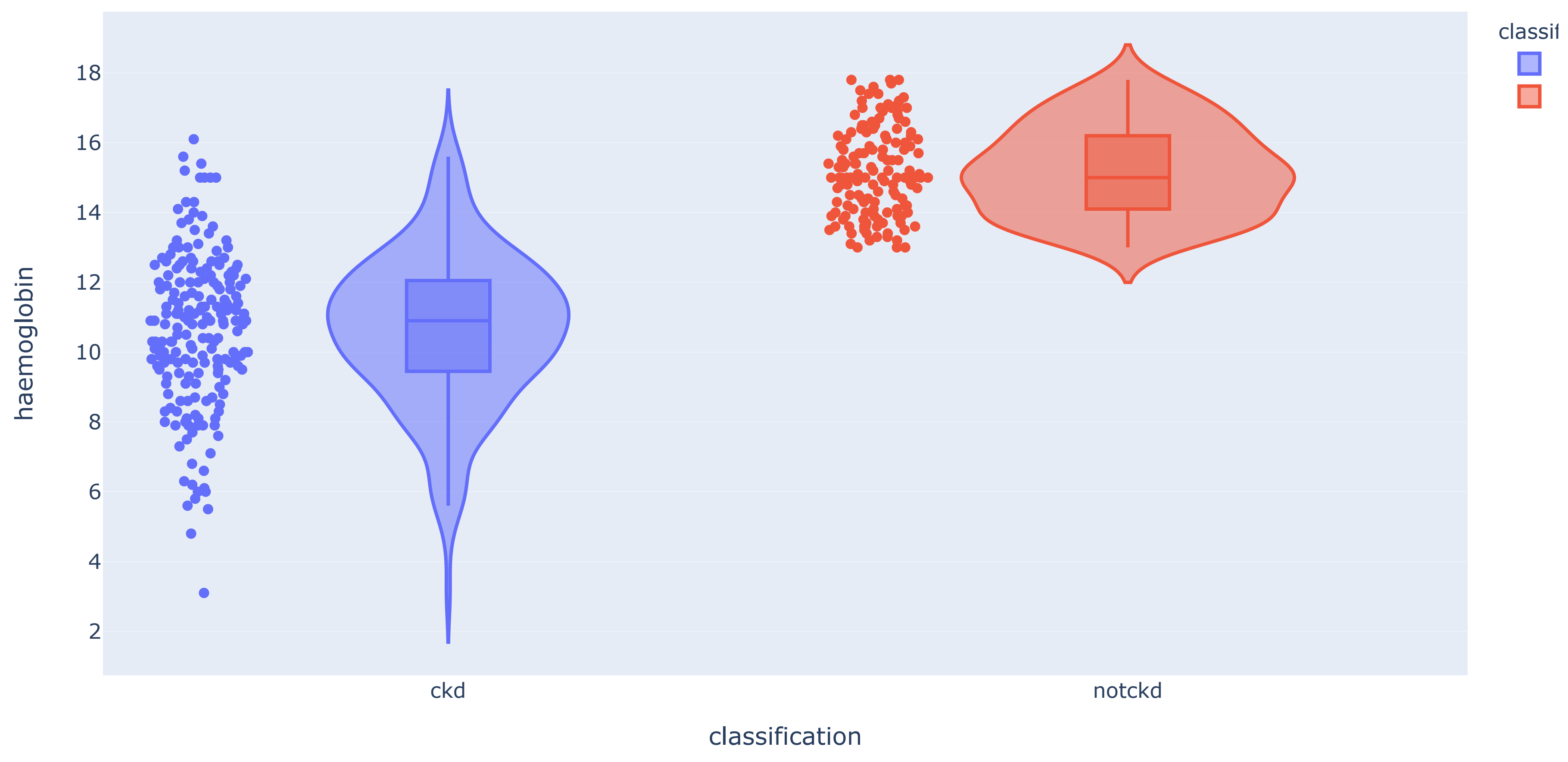


One outlier seems to be there. But can't assert that it really is an outlier. Hence won't drop

```
In [193]: violin('packed_cell_volume')
```

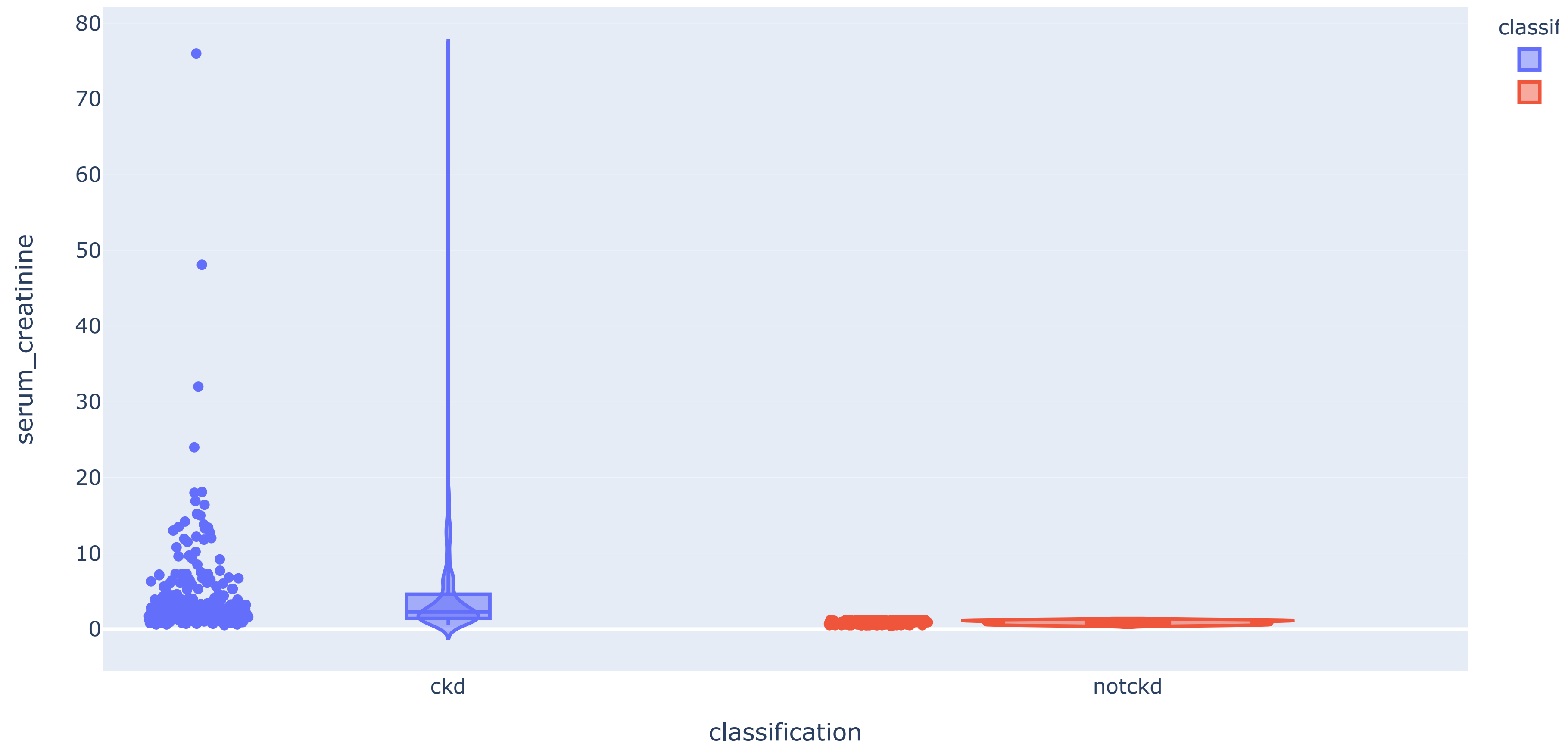


```
In [195]: violin('haemoglobin')
```



These seem to be good

```
In [198]: violin('serum_creatinine')
```

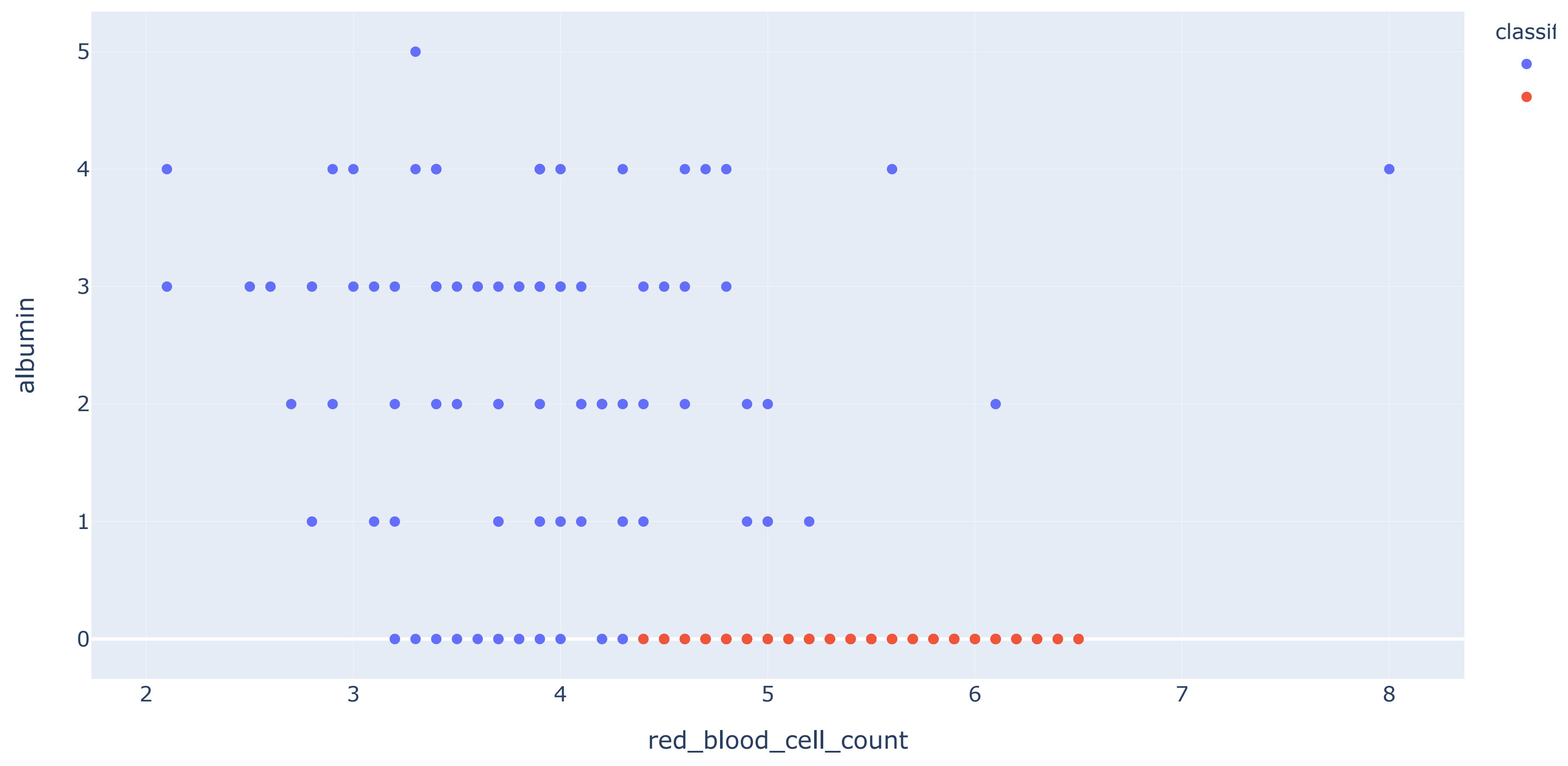


Serum creatinine has got about 2 seemingly outliers

7.2 Now let's check for negative correlation and its impact on classes

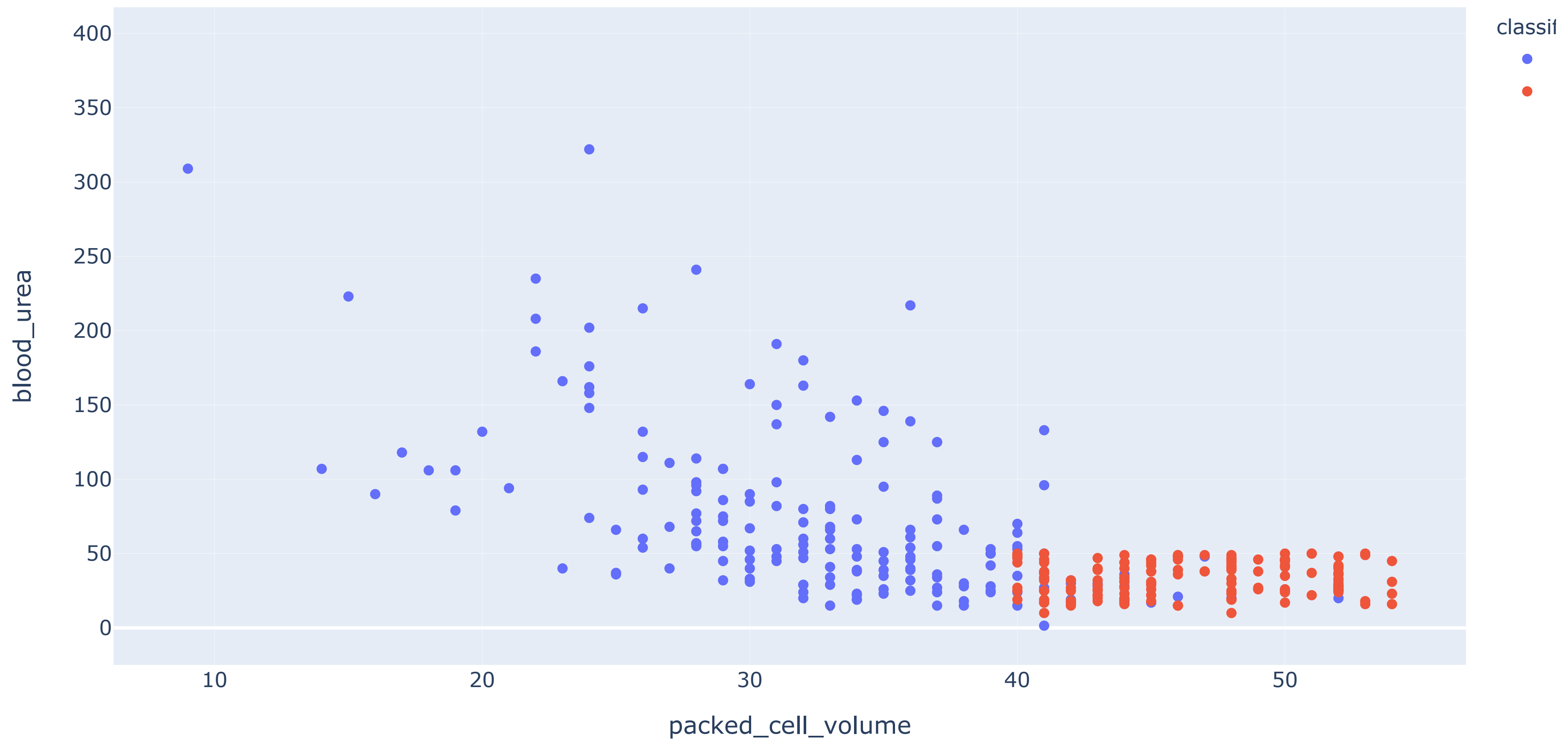
```
Albumin, Blood urea -> Red blood cell count, packed cell volume, Haemoglobin
```

In [215]:
```
scatters('red_blood_cell_count','albumin')
```



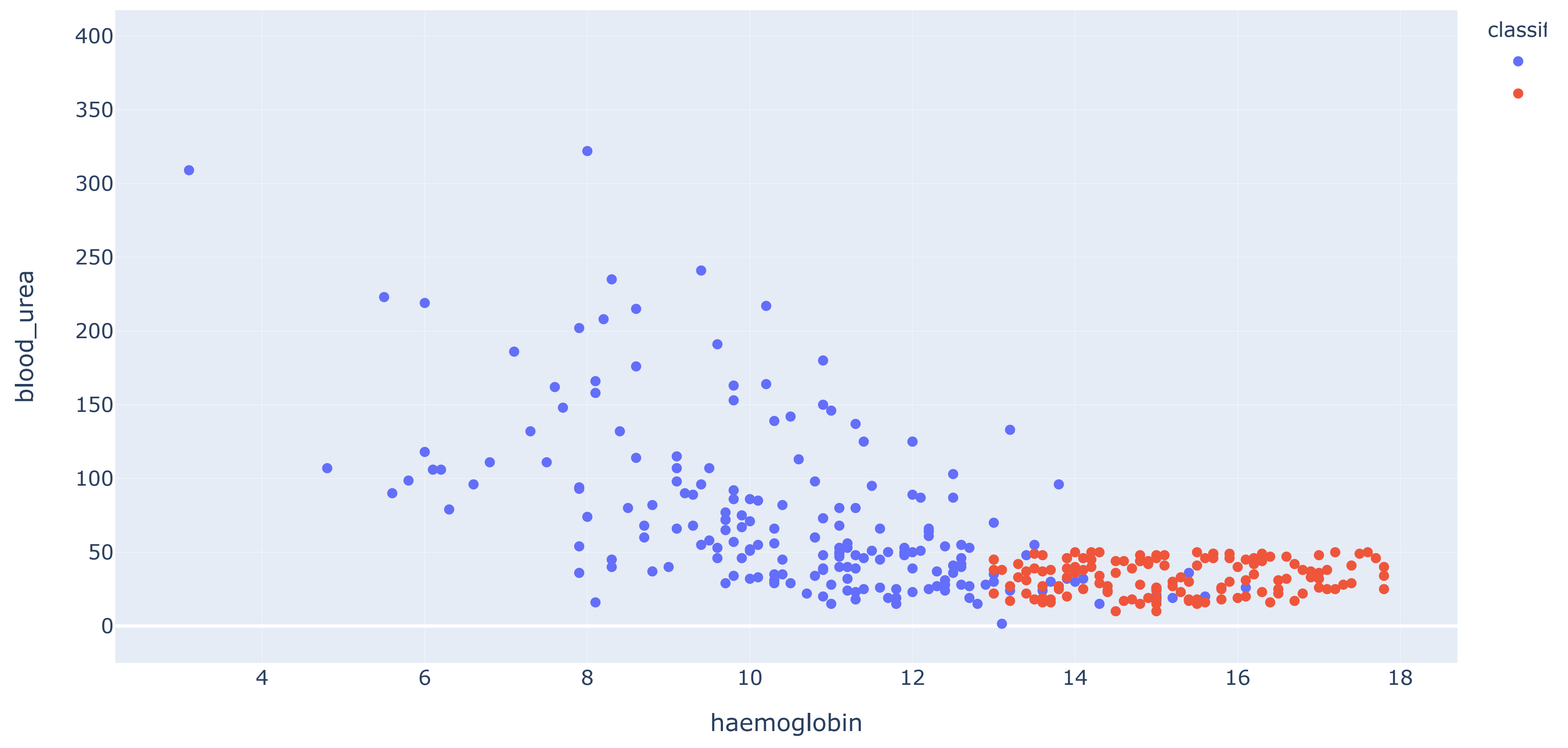
Clearly, albumin levels of above 0 affect ckd largely

In [225]:
```
scatters('packed_cell_volume','blood_urea')
```

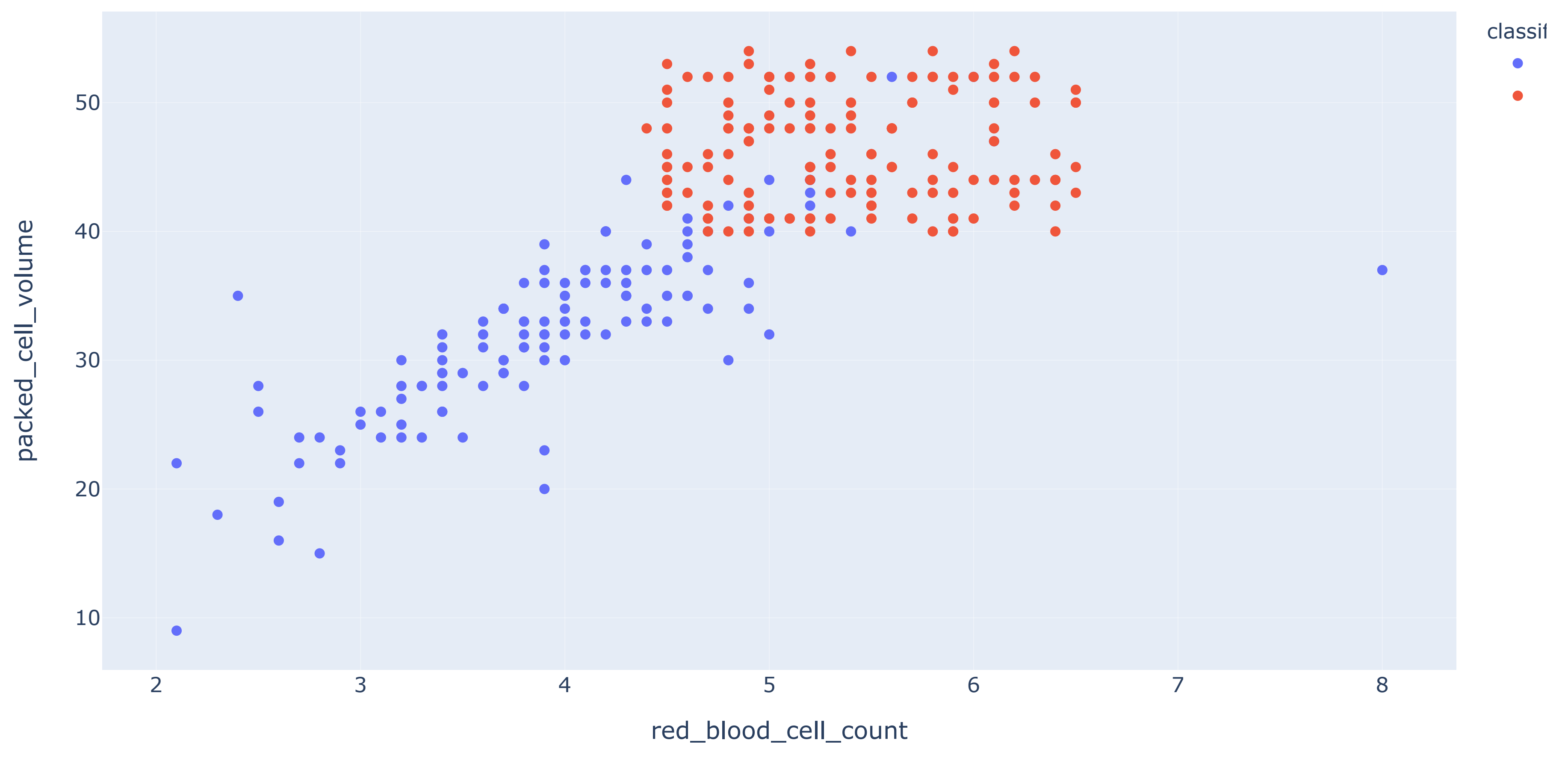


Packed cell volume >= 40 largely affects to be non ckd

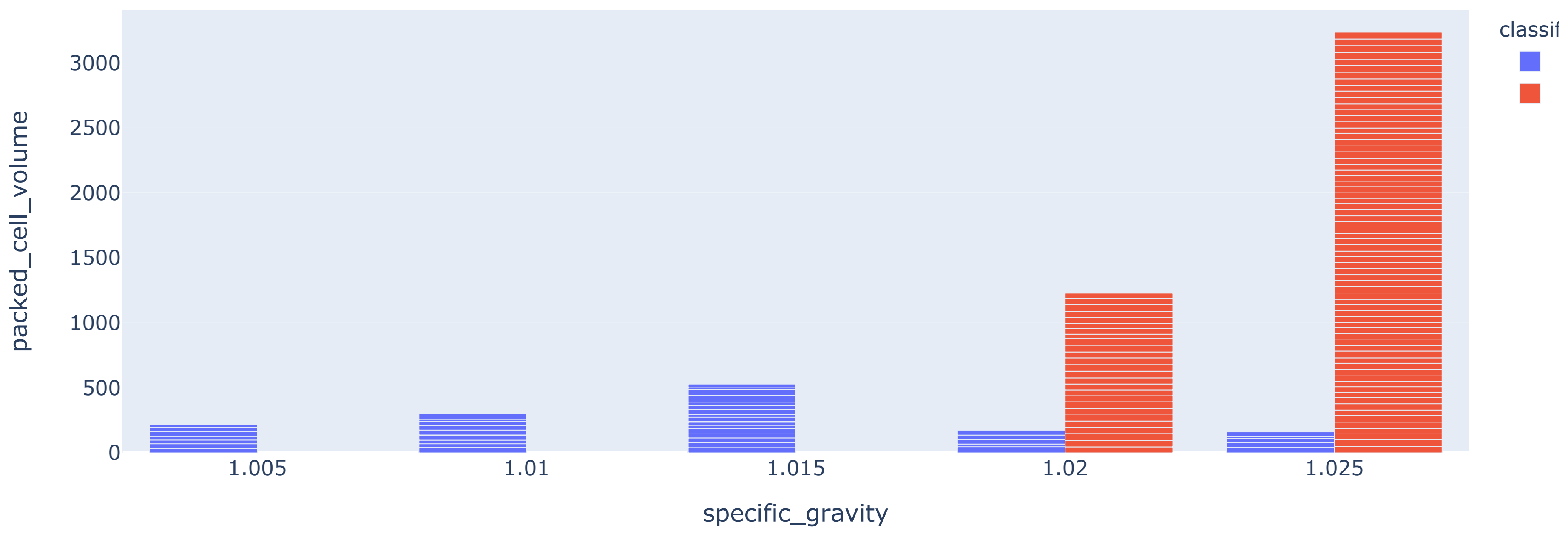
```
In [228]: scatters('haemoglobin','blood_urea')
```



```
In [230]: scatters('red_blood_cell_count','packed_cell_volume')
```

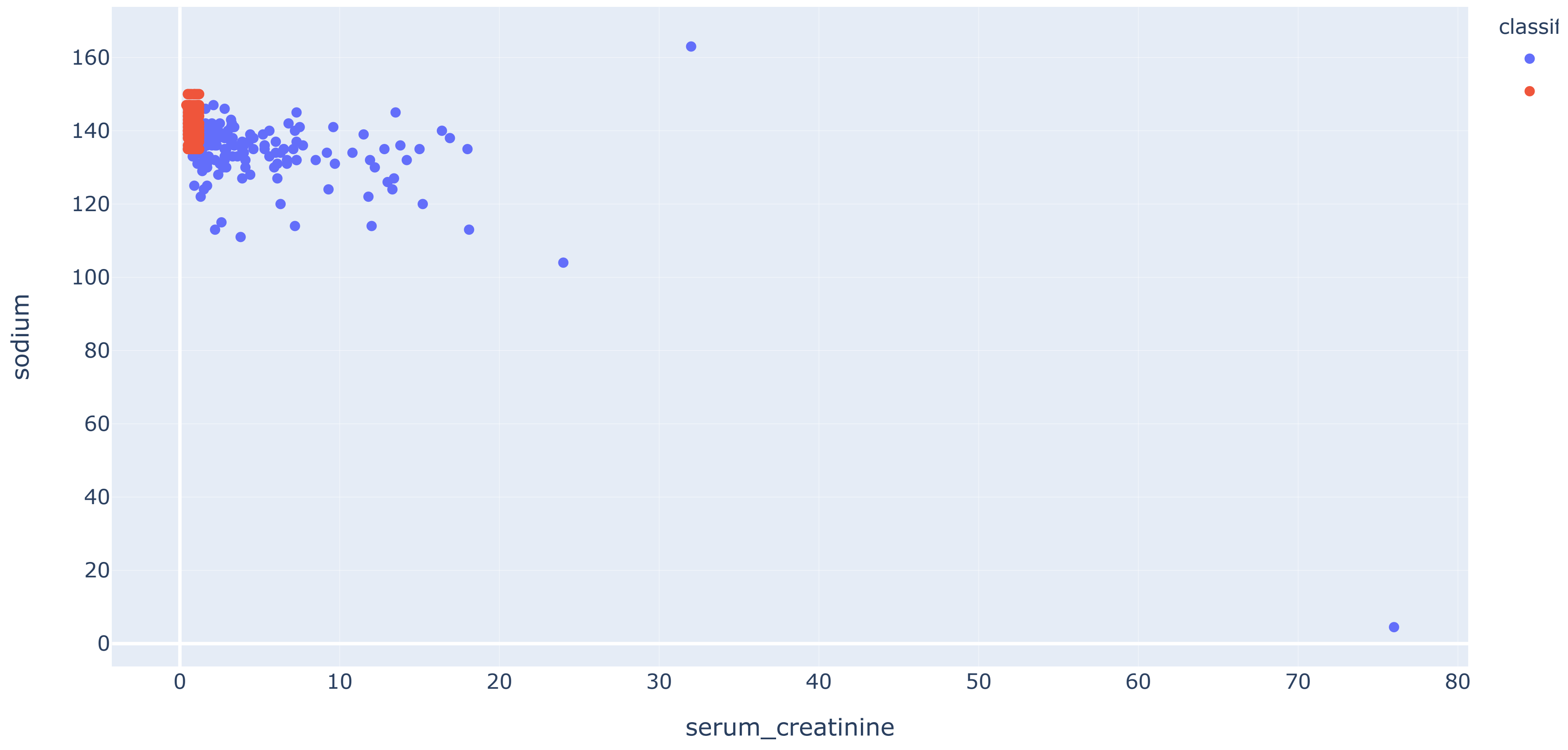


```
In [232]: fig = px.bar(df,
                       x="specific_gravity",
                       y="packed_cell_volume",
                       color='classification',
                       barmode='group', height=400)
          fig.show()
```



Clearly, specific gravity >=1.02 affects non ckd

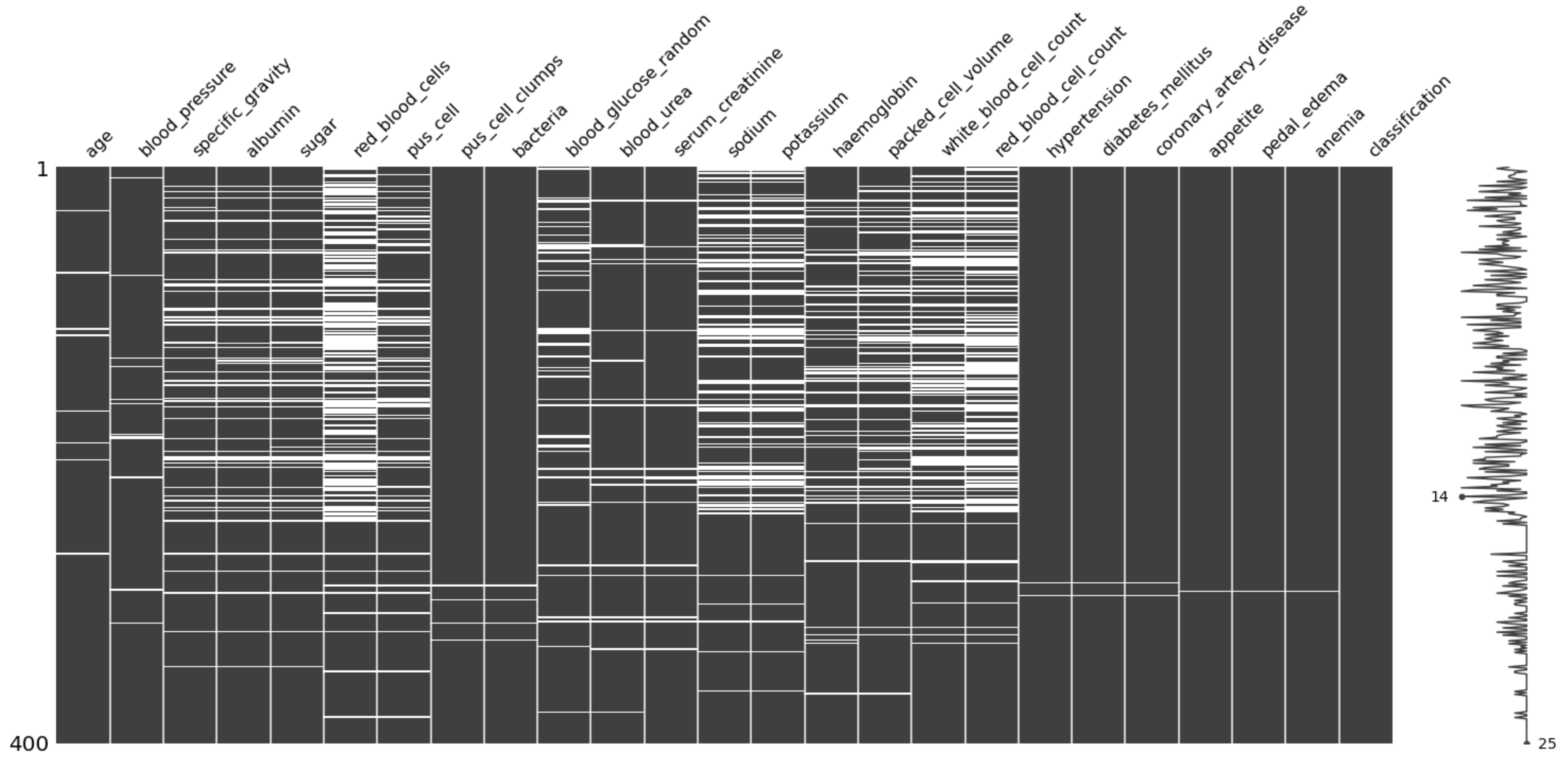
```
In [240]: # Sodium and serum_creatinine

          scatters('serum_creatinine','sodium')
```



Insights, sodium above 135 and serum creatinine less than 1.3 are non ckd

```
In [244]: import missingno as msno
          %matplotlib inline
```

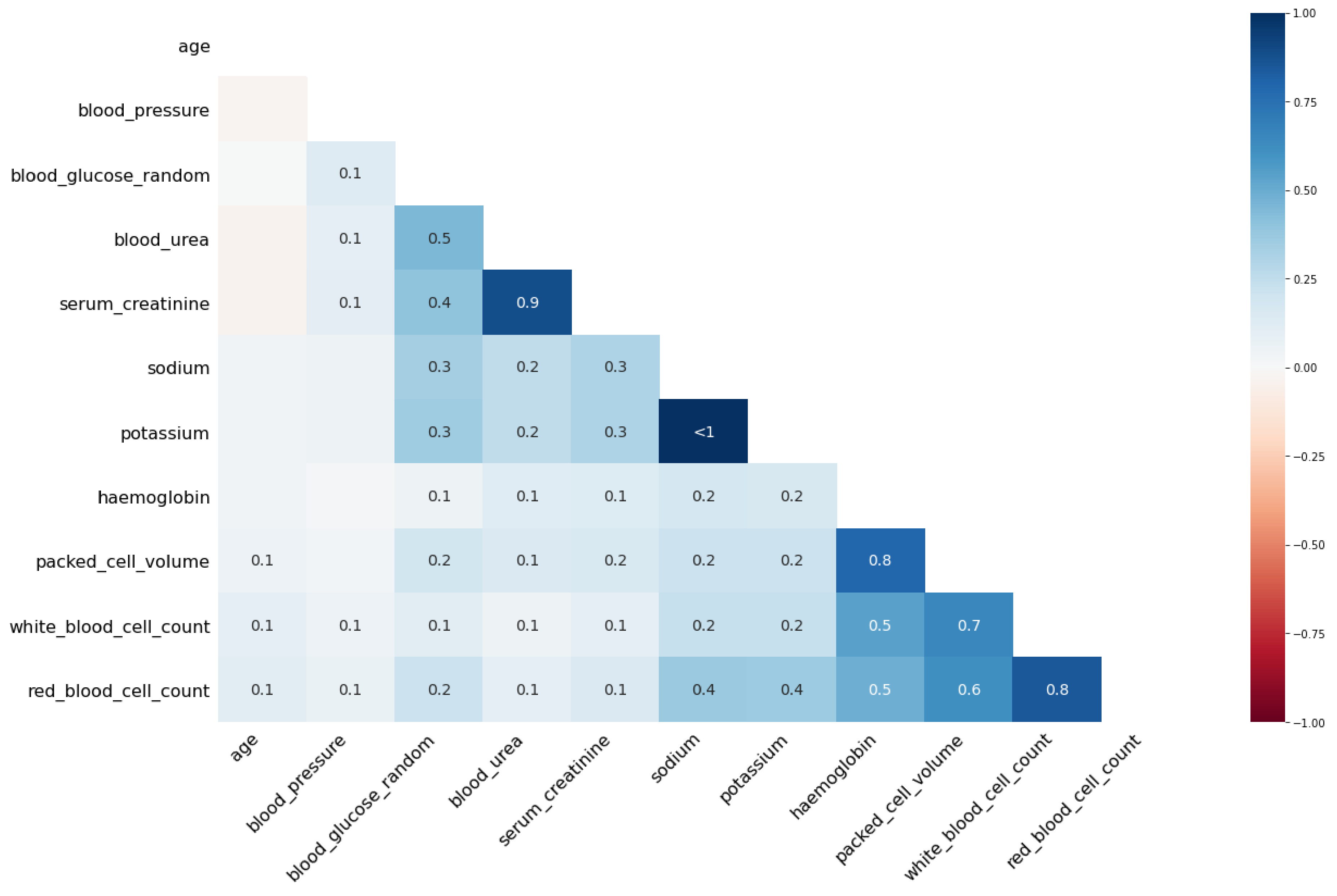
```
In [271]: fig = msno.matrix(df)
```



Missingno is a Python library that provides the ability to understand the distribution of missing values through informative visualizations.

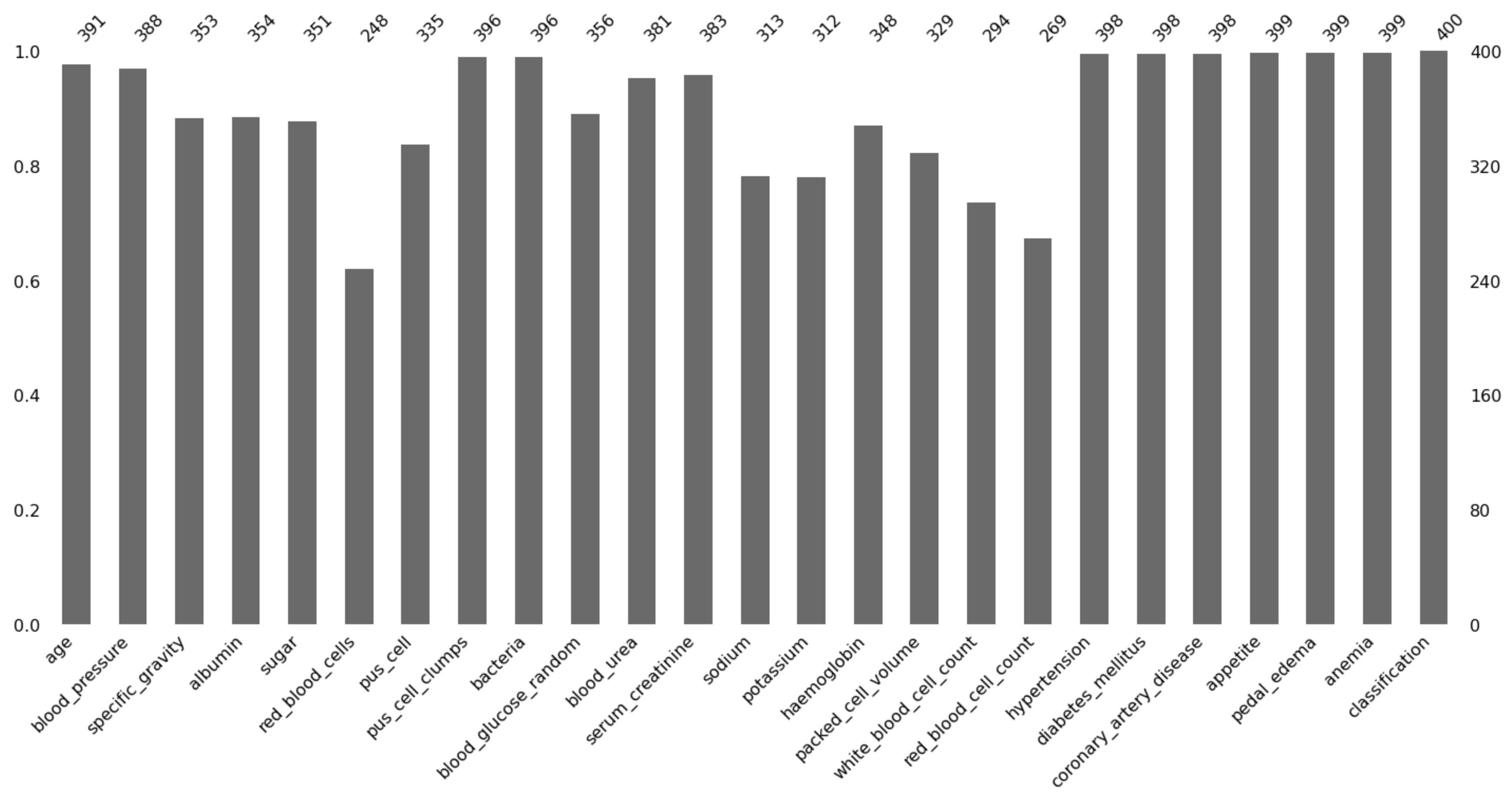
The visualizations can be in the form of heat maps and bar charts.

```
In [288]: msno.heatmap(df[numerical_features])
Out[288]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb841e71400>
```



In [303]:
```
msno.bar(df)
```

Out[303]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8361bd820>
```



8. Summary:

- RBC count is highly correlated with hemoglobin and packed_volume_area
- red blood cell level is abnormal and RBC count high or low than a normal range than person likely to have CKD
- white_blood_cell, sodium, and potassium, blood_press no such relation with other variables
- if a person is having less specific gravity(levels 1.005,1.01,1.015) with lesser hemoglobin(<13) and less packed cell volume(<40) higher chances of having CKD
- Albumin and Hemoglobin have a negative correlation(correlation matric) mostly albumin level above 0 and hemoglobin higher than normal range is an indication CKD
- If blood urea level is higher than 150 than there are higher chances of having a chronic kidney disease
- Higher the serum creatinine level i.e >1.2, people likely to have chronic kidney diseases.
- People who had blood pressure <60 to >80 are prone to have a chronic disease.
- high range of blood glucose random, sugar level above 1and also suffering from diabetes_mellitus are majorly classified as chronic kidney disease.
- Age has no such correlation with other variables
- red_blood_cell - red_blood_cell_count
- From the above analysis, we can see that presence of even one - abnormal red cell count, bacteria, hypertension, pus cells, diabetes, coronary disease, lack of appetite, amenic increases chances of occurence of ckd to substantial level.

Conclusion

Supervised machine learning helps you to solve various types of Chronic Kidney Disease problems. Classification technique under supervised learning are suitable for our Chronic Kidney Disease dataset. Outputs always have a probabilistic interpretation, and the algorithm can be regularized to avoid overfitting.

In [ ]: