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

In [2]:
```
df = pd.read_csv("kidney_disease.csv")
```

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

Out[3]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no

5 rows × 26 columns

In [4]:
```
df.columns = ['id','age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'r
              'pus_cell_clumps', 'bacteria', 'blood_glucose_random', 'blood_urea', 'se
              'potassium', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_cou
              'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appeti
              'anaemia', 'class']
columns = ['id','age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'red
           'pus_cell_clumps', 'bacteria', 'blood_glucose_random', 'blood_urea', 'se
           'potassium', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_cou
           'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appeti
           'anaemia', 'class']
```

In [5]:
```
df.describe()
```

Out[5]:

	id	age	blood_pressure	specific_gravity	albumin	sugar	blood_glucose_random
count	400.000000	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000
mean	199.500000	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517
std	115.614301	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714
min	0.000000	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000
25%	99.750000	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000
50%	199.500000	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000
75%	299.250000	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000
max	399.000000	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
id                          0
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    105
red_blood_cell_count      130
hypertension                2
diabetes_mellitus           2
coronary_artery_disease     2
appetite                    1
pedal_edema                 1
anaemia                     1
class                       0
dtype: int64
```

In [7]:
```
df.info()
```

Out[7]:
```
<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  anaemia                  399 non-null    object
 25  class                    400 non-null    object
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

In [8]:
```
for column in columns:
    mode_value = df[column].mode()[0]  # Calculate the mode for the column
    df[column].fillna(mode_value, inplace=True)
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
id                         0
age                        0
blood_pressure             0
specific_gravity           0
albumin                    0
sugar                      0
red_blood_cells            0
pus_cell                   0
pus_cell_clumps            0
bacteria                   0
blood_glucose_random       0
blood_urea                 0
serum_creatinine           0
sodium                     0
potassium                  0
haemoglobin                0
packed_cell_volume         0
white_blood_cell_count     0
red_blood_cell_count       0
hypertension               0
diabetes_mellitus          0
coronary_artery_disease    0
appetite                   0
pedal_edema                0
anaemia                    0
class                      0
dtype: int64
```

In [10]:
```
df.drop(['red_blood_cells','pus_cell','pus_cell_clumps','bacteria'],inplace = True, a
```

In [11]:
```
df['packed_cell_volume'] = df['packed_cell_volume'].replace("\t?", pd.NA)

# Convert the column to integers (assuming the column contains valid numeric strings)
df['packed_cell_volume'] = pd.to_numeric(df['packed_cell_volume'], errors='coerce').a
```

In [12]:
```
df['white_blood_cell_count'] = df['white_blood_cell_count'].replace("\t?", pd.NA)

# Convert the column to integers (assuming the column contains valid numeric strings)
df['white_blood_cell_count'] = pd.to_numeric(df['white_blood_cell_count'], errors='co
```

In [13]:
```
df.info()
```

Out[13]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 22 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       400 non-null    int64
 1   age                      400 non-null    float64
 2   blood_pressure           400 non-null    float64
 3   specific_gravity         400 non-null    float64
 4   albumin                  400 non-null    float64
 5   sugar                    400 non-null    float64
 6   blood_glucose_random     400 non-null    float64
 7   blood_urea               400 non-null    float64
 8   serum_creatinine         400 non-null    float64
 9   sodium                   400 non-null    float64
 10  potassium                400 non-null    float64
 11  haemoglobin              400 non-null    float64
 12  packed_cell_volume       400 non-null    object
 13  white_blood_cell_count   399 non-null    object
 14  red_blood_cell_count     400 non-null    object
 15  hypertension             400 non-null    object
 16  diabetes_mellitus        400 non-null    object
 17  coronary_artery_disease  400 non-null    object
 18  appetite                 400 non-null    object
 19  pedal_edema              400 non-null    object
 20  anaemia                  400 non-null    object
 21  class                    400 non-null    object
dtypes: float64(11), int64(1), object(10)
memory usage: 65.8+ KB
```

In [14]:
```
df['sugar'].value_counts()
```

Out[14]:
```
0.0    339
2.0     18
1.0     14
3.0     13
4.0     11
5.0      3
Name: sugar, dtype: int64
```

In [15]:
```
df['red_blood_cell_count'] = df['red_blood_cell_count'].replace("\t?", pd.NA)

# Convert the column to integers (assuming the column contains valid numeric strings)
df['red_blood_cell_count'] = pd.to_numeric(df['red_blood_cell_count'], errors='coerce
```

In [16]:
```
columns1 = ['id','age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar','blo
            'potassium', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_cou
            'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appeti
            'anaemia', 'class']
for column in columns1:
    mode_value = df[column].mode()[0]  # Calculate the mode for the column
    df[column].fillna(mode_value, inplace=True)
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder
```

In [18]:
```
label_encoder = LabelEncoder()
df['hypertension'] = label_encoder.fit_transform(df['hypertension'])
df['diabetes_mellitus'] = label_encoder.fit_transform(df['diabetes_mellitus'])
df['coronary_artery_disease'] = label_encoder.fit_transform(df['coronary_artery_disea
df['appetite'] = label_encoder.fit_transform(df['appetite'])
df['pedal_edema'] = label_encoder.fit_transform(df['pedal_edema'])
df['anaemia'] = label_encoder.fit_transform(df['anaemia'])
```

In [19]:
```
def remove_outliers_iqr(df):
    # Calculate the first quartile (Q1) and third quartile (Q3)
    q1 = np.percentile(df, 25)
    q3 = np.percentile(df, 75)

    # Calculate the interquartile range (IQR)
    iqr = q3 - q1

    # Define the lower and upper bounds for outliers
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    # Remove outliers
    df = df[(df >= lower_bound) & (df <= upper_bound)]

    return df
```

In [20]:
```
df.drop(['id','specific_gravity','albumin','appetite','anaemia','blood_urea','serum_c
         'sodium','potassium','white_blood_cell_count','coronary_artery_disease','peda
```

In [ ]:

In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [22]:
```
plt.figure(figsize=(20,16))
sns.heatmap(df.corr(), annot=True, fmt='.2f')
```

Out[22]:
```
<AxesSubplot:>
```

In [23]:
```
from sklearn.model_selection import train_test_split
X = df.drop('class', axis = 1)
y = df['class']
```

In [24]:
```
from sklearn.preprocessing import MinMaxScaler
scalar = MinMaxScaler()
```

In [25]:
```
scalar.fit(X)
standardised_data = scalar.transform(X)
```

In [26]:
```
X = standardised_data
y = df['class']
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=
```

In [28]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=20, criterion='entropy', random_state=42)
```

In [29]:
```
rf.fit(X_train, y_train)
```

Out[29]:
```
RandomForestClassifier(criterion='entropy', n_estimators=20, random_state=42)
```

In [30]:
```
y_pred = rf.predict(X_test)
```

In [31]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[31]:
```
0.9625
```

In [32]:
```
df
```

Out[32]:

	age	blood_pressure	haemoglobin	hypertension	diabetes_mellitus	coronary_artery_disease	class
0	48.0	80.0	15.4	1	4	1	ckd
1	7.0	50.0	11.3	0	3	1	ckd
2	62.0	80.0	9.6	0	4	1	ckd
3	48.0	70.0	11.2	1	3	1	ckd
4	51.0	80.0	11.6	0	3	1	ckd
...	...	...	...	...	...	...	...
395	55.0	80.0	15.7	0	3	1	notckd
396	42.0	70.0	16.5	0	3	1	notckd
397	12.0	80.0	15.8	0	3	1	notckd
398	17.0	60.0	14.2	0	3	1	notckd
399	58.0	80.0	15.8	0	3	1	notckd

400 rows × 7 columns

In [ ]:
```
df
```