Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import joblib
import pickle
%matplotlib inline
data=pd.read_csv('chronickidneydisease.csv')
data.head()
#   UNDERSTANDING THE DATATYPE AND SUMMARY
 data.dtypes
data.describe()
data.info()
# HANDLING MISSING VALUES
data=data.drop("id",axis='columns')
data
data.to_csv('ckd.csv',sep=',',index=False)
data=pd.read_csv('ckd.csv')
data
data = pd.read_csv("ckd.csv",header=0, na_values="?")
data = pd.read_csv("ckd.csv",header=0, na_values="\t?")
data['classification'] = data['classification'].map({'ckd':1,'notckd':0,'continuous':0})
data['htn'] = data['htn'].map({'yes':1,'no':0})
data['dm'] = data['dm'].map({'yes':1,'no':0})
data['cad'] = data['cad'].map({'yes':1,'no':0})
data['appet'] = data['appet'].map({'good':1,'poor':0})
data['ane'] = data['ane'].map({'yes':1,'no':0})
data['pe'] = data['pe'].map({'yes':1,'no':0})
```

```
data['ba'] = data['ba'].map({'present':1,'notpresent':0})
data['pcc'] = data['pcc'].map({'present':1,'notpresent':0})
data['pc'] = data['pc'].map({'abnormal':1,'normal':0})
data['rbc'] = data['rbc'].map({'abnormal':1,'normal':0})
# REPLACING THE MISSING VALUES
data.replace("?", np.NaN)
data.replace("\t?", np.NaN)
data
data.fillna(round(data.mean(),2), inplace=True)
data
data = data[np.isfinite(data).all(1)]
data.fillna(round(data.mean(),2), inplace=True)
#data.fillna(data.mean())
#data.isnull().sum()
#mv=data['rc'].mean()
#data = data.groupby(data.columns, axis = 1).transform(lambda x: x.fillna(x.mean()))
#fill_mean = lambda col : col.fillna(col.mean())
data=data.apply(lambda x:x.fillna(x.value_counts().index[0]))
data
data
data.shape[0], data.dropna().shape[0]
data.columns
data.to_csv("ckd_pp.csv", sep=',', index=False)
data=pd.read_csv("ckd_pp.csv")
data
data['classification'].value_counts()
plt.figure(figsize = (19,19))
sns.heatmap(data.corr(), annot = True, cmap = 'coolwarm')
data.shape
data.isnull().sum()
plt.hist(data['classification'])
plt.title('classification(ckd=1 , notckd=0)')
```

```
plt.show()

#Plotting Correletion matrix

plt.figure(figsize=(19, 19))

plt.title('CKD Attributes Correlation')

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')  # looking for strong correlations with "class" row

plt.show()

# LABEL ENCODING

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

# SPLITTING DATASET INTO DEPENDENT AND INDEPENDENT VARIABLE

x = data.iloc[:,:-1]

y = data['classification']

lab_enc = preprocessing.LabelEncoder()

y = lab_enc.fit_transform(y)

# SPLITTING THE DATASET INTO TRAINING AND TEST TEST

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.31, random_state =2029)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

# MODEL BUILDING

logreg = LogisticRegression(solver='lbfgs',max_iter=10000)

logreg.fit(X_train,y_train)

# TESTING THE MODEL

test_pred = logreg.predict(X_test)

train_pred = logreg.predict(X_train)

import sklearn.metrics as metrics

# EVALUATING THE MODEL

from sklearn.metrics import accuracy_score, confusion_matrix

print('Train Accuracy: ', accuracy_score(y_train, train_pred))

print('Test Accuracy: ', accuracy_score(y_test, test_pred))

# SAVE THE MODEL
```

```
filename = "Completed_model.joblib"
  joblib.dump(logreg, filename)
loaded_model = joblib.load(filename)
result = loaded_model.score(X_test, y_test)
print(result)
```

Screenshots of Machine Learning code:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

import joblib
import pickle

%matplotlib inline
```

In [2]:
```
data=pd.read_csv('chronickidneydisease.csv')
```

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

Out[3]:

	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

UNDERSTANDING THE DATATYPE AND SUMMARY

In [4]:
```
data.dtypes
```

Out[4]:
```
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
```

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

Out[5]:

	id	age	bp	sg	al	su	bgr	bu	sc	sod	pot	hemo
count	400.000000	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000
mean	199.500000	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437
std	115.614301	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587
min	0.000000	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000
25%	99.750000	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000
50%	199.500000	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000
75%	299.250000	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000
max	399.000000	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000

In [6]:
```
data.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
```

HANDLING MISSING VALUES

In [7]:
```
data=data.drop("id",axis='columns')
```

In [8]:
```
data
```

Out[8]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	140.0	...	47	6700	4.9	no	no	no	good	no	no	notckd
396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	75.0	...	54	7800	6.2	no	no	no	good	no	no	notckd
397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	100.0	...	49	6600	5.4	no	no	no	good	no	no	notckd
398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	114.0	...	51	7200	5.9	no	no	no	good	no	no	notckd
399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	131.0	...	53	6800	6.1	no	no	no	good	no	no	notckd

400 rows × 25 columns

In [9]:
```
data.to_csv('ckd.csv',sep=',',index=False)
```

In [10]:
```
data=pd.read_csv('ckd.csv')
```

In [11]:
```
data
```

Out[11]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	140.0	...	47	6700	4.9	no	no	no	good	no	no	notckd
396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	75.0	...	54	7800	6.2	no	no	no	good	no	no	notckd
397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	100.0	...	49	6600	5.4	no	no	no	good	no	no	notckd
398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	114.0	...	51	7200	5.9	no	no	no	good	no	no	notckd
399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	131.0	...	53	6800	6.1	no	no	no	good	no	no	notckd

400 rows × 25 columns

In [12]:
```
data = pd.read_csv("ckd.csv",header=0, na_values="?")
```

In [13]:
```
data = pd.read_csv("ckd.csv",header=0, na_values="\t?")
```

In [14]:
```
data['classification'] = data['classification'].map({'ckd':1,'notckd':0,'continuous':0})
data['htn'] = data['htn'].map({'yes':1,'no':0})
data['dm'] = data['dm'].map({'yes':1,'no':0})
data['cad'] = data['cad'].map({'yes':1,'no':0})
data['appet'] = data['appet'].map({'good':1,'poor':0})
data['ane'] = data['ane'].map({'yes':1,'no':0})
data['pe'] = data['pe'].map({'yes':1,'no':0})
data['ba'] = data['ba'].map({'present':1,'notpresent':0})
data['pcc'] = data['pcc'].map({'present':1,'notpresent':0})
data['pc'] = data['pc'].map({'abnormal':1,'normal':0})
data['rbc'] = data['rbc'].map({'abnormal':1,'normal':0})
```

REPLACING THE MISSING VALUES

In [15]:
```
data.replace("?", np.NaN)
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	0.0	0.0	0.0	121.0	...	44.0	7800.0	5.2	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1	7.0	50.0	1.020	4.0	0.0	NaN	0.0	0.0	0.0	NaN	...	38.0	6000.0	NaN	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	...	31.0	7500.0	NaN	0.0	1.0	0.0	0.0	0.0	1.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	32.0	6700.0	3.9	1.0	0.0	0.0	0.0	1.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	...	35.0	7300.0	4.6	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	140.0	...	47.0	6700.0	4.9	0.0	0.0	0.0	1.0	0.0	0.0	0.0
396	42.0	70.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	75.0	...	54.0	7800.0	6.2	0.0	0.0	0.0	1.0	0.0	0.0	0.0
397	12.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	100.0	...	49.0	6600.0	5.4	0.0	0.0	0.0	1.0	0.0	0.0	0.0
398	17.0	60.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	114.0	...	51.0	7200.0	5.9	0.0	0.0	0.0	1.0	0.0	0.0	0.0
399	58.0	80.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	131.0	...	53.0	6800.0	6.1	0.0	0.0	0.0	1.0	0.0	0.0	0.0

400 rows × 25 columns

In [16]:
```
data.replace("\t?", np.NaN)
```

Out[16]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	0.0	0.0	0.0	121.0	...	44.0	7800.0	5.2	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1	7.0	50.0	1.020	4.0	0.0	NaN	0.0	0.0	0.0	NaN	...	38.0	6000.0	NaN	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	...	31.0	7500.0	NaN	0.0	1.0	0.0	0.0	0.0	1.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	32.0	6700.0	3.9	1.0	0.0	0.0	0.0	1.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	...	35.0	7300.0	4.6	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	140.0	...	47.0	6700.0	4.9	0.0	0.0	0.0	1.0	0.0	0.0	0.0
396	42.0	70.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	75.0	...	54.0	7800.0	6.2	0.0	0.0	0.0	1.0	0.0	0.0	0.0
397	12.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	100.0	...	49.0	6600.0	5.4	0.0	0.0	0.0	1.0	0.0	0.0	0.0
398	17.0	60.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	114.0	...	51.0	7200.0	5.9	0.0	0.0	0.0	1.0	0.0	0.0	0.0
399	58.0	80.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	131.0	...	53.0	6800.0	6.1	0.0	0.0	0.0	1.0	0.0	0.0	0.0

400 rows × 25 columns

In [17]:
```
data
```

Out[17]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	0.0	0.0	0.0	121.0	...	44.0	7800.0	5.2	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1	7.0	50.0	1.020	4.0	0.0	NaN	0.0	0.0	0.0	NaN	...	38.0	6000.0	NaN	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	...	31.0	7500.0	NaN	0.0	1.0	0.0	0.0	0.0	1.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	32.0	6700.0	3.9	1.0	0.0	0.0	0.0	1.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	...	35.0	7300.0	4.6	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	140.0	...	47.0	6700.0	4.9	0.0	0.0	0.0	1.0	0.0	0.0	0.0

In [18]:
```
data.fillna(round(data.mean(),2), inplace=True)
```

In [19]:
```
data
```

Out[19]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	0.19	0.0	0.0	0.0	121.00	...	44.0	7800.0	5.20	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1	7.0	50.0	1.020	4.0	0.0	0.19	0.0	0.0	0.0	148.04	...	38.0	6000.0	4.71	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	62.0	80.0	1.010	2.0	3.0	0.00	0.0	0.0	0.0	423.00	...	31.0	7500.0	4.71	0.0	1.0	0.0	0.0	0.0	1.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.00	1.0	1.0	0.0	117.00	...	32.0	6700.0	3.90	1.0	0.0	0.0	0.0	1.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.00	0.0	0.0	0.0	106.00	...	35.0	7300.0	4.60	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	0.00	0.0	0.0	0.0	140.00	...	47.0	6700.0	4.90	0.0	0.0	0.0	1.0	0.0	0.0	0.0
396	42.0	70.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	75.00	...	54.0	7800.0	6.20	0.0	0.0	0.0	1.0	0.0	0.0	0.0
397	12.0	80.0	1.020	0.0	0.0	0.00	0.0	0.0	0.0	100.00	...	49.0	6600.0	5.40	0.0	0.0	0.0	1.0	0.0	0.0	0.0
398	17.0	60.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	114.00	...	51.0	7200.0	5.90	0.0	0.0	0.0	1.0	0.0	0.0	0.0
399	58.0	80.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	131.00	...	53.0	6800.0	6.10	0.0	0.0	0.0	1.0	0.0	0.0	0.0

400 rows × 25 columns

In [20]:
```
data = data[np.isfinite(data).all(1)]
```

In [21]:
```
data.fillna(round(data.mean(),2), inplace=True)
```

In [22]:
```
#data.fillna(data.mean())
#data.isnull().sum()
#mv=data['rc'].mean()
#data = data.groupby(data.columns, axis = 1).transform(lambda x: x.fillna(x.mean()))
#fill_mean = lambda col : col.fillna(col.mean())
data=data.apply(lambda x:x.fillna(x.value_counts().index[0]))
```

In [23]:
```
data
```

Out[23]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	0.19	0.0	0.0	0.0	121.00	...	44.0	7800.0	5.20	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1	7.0	50.0	1.020	4.0	0.0	0.19	0.0	0.0	0.0	148.04	...	38.0	6000.0	4.71	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	62.0	80.0	1.010	2.0	3.0	0.00	0.0	0.0	0.0	423.00	...	31.0	7500.0	4.71	0.0	1.0	0.0	0.0	0.0	1.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.00	1.0	1.0	0.0	117.00	...	32.0	6700.0	3.90	1.0	0.0	0.0	0.0	1.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.00	0.0	0.0	0.0	106.00	...	35.0	7300.0	4.60	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	0.00	0.0	0.0	0.0	140.00	...	47.0	6700.0	4.90	0.0	0.0	0.0	1.0	0.0	0.0	0.0
396	42.0	70.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	75.00	...	54.0	7800.0	6.20	0.0	0.0	0.0	1.0	0.0	0.0	0.0
397	12.0	80.0	1.020	0.0	0.0	0.00	0.0	0.0	0.0	100.00	...	49.0	6600.0	5.40	0.0	0.0	0.0	1.0	0.0	0.0	0.0
398	17.0	60.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	114.00	...	51.0	7200.0	5.90	0.0	0.0	0.0	1.0	0.0	0.0	0.0
399	58.0	80.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	131.00	...	53.0	6800.0	6.10	0.0	0.0	0.0	1.0	0.0	0.0	0.0

400 rows × 25 columns

In [25]:
```
data.shape[0], data.dropna().shape[0]
```

Out[25]:
```
(400, 400)
```

In [26]:
```
data.columns
```

Out[26]:
```
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')
```

In [27]:
```
data.to_csv("ckd_pp.csv", sep=',', index=False)
```

In [28]:
```
data=pd.read_csv("ckd_pp.csv")
```

In [29]:
```
data
```

Out[29]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	0.19	0.0	0.0	0.0	121.00	...	44.0	7800.0	5.20	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1	7.0	50.0	1.020	4.0	0.0	0.19	0.0	0.0	0.0	148.04	...	38.0	6000.0	4.71	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	62.0	80.0	1.010	2.0	3.0	0.00	0.0	0.0	0.0	423.00	...	31.0	7500.0	4.71	0.0	1.0	0.0	0.0	0.0	1.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.00	1.0	1.0	0.0	117.00	...	32.0	6700.0	3.90	1.0	0.0	0.0	0.0	1.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.00	0.0	0.0	0.0	106.00	...	35.0	7300.0	4.60	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	0.00	0.0	0.0	0.0	140.00	...	47.0	6700.0	4.90	0.0	0.0	0.0	1.0	0.0	0.0	0.0
396	42.0	70.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	75.00	...	54.0	7800.0	6.20	0.0	0.0	0.0	1.0	0.0	0.0	0.0
397	12.0	80.0	1.020	0.0	0.0	0.00	0.0	0.0	0.0	100.00	...	49.0	6800.0	5.40	0.0	0.0	0.0	1.0	0.0	0.0	0.0
398	17.0	60.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	114.00	...	51.0	7200.0	5.90	0.0	0.0	0.0	1.0	0.0	0.0	0.0
399	58.0	80.0	1.025	0.0	0.0	0.00	0.0	0.0	0.0	131.00	...	53.0	6800.0	6.10	0.0	0.0	0.0	1.0	0.0	0.0	0.0

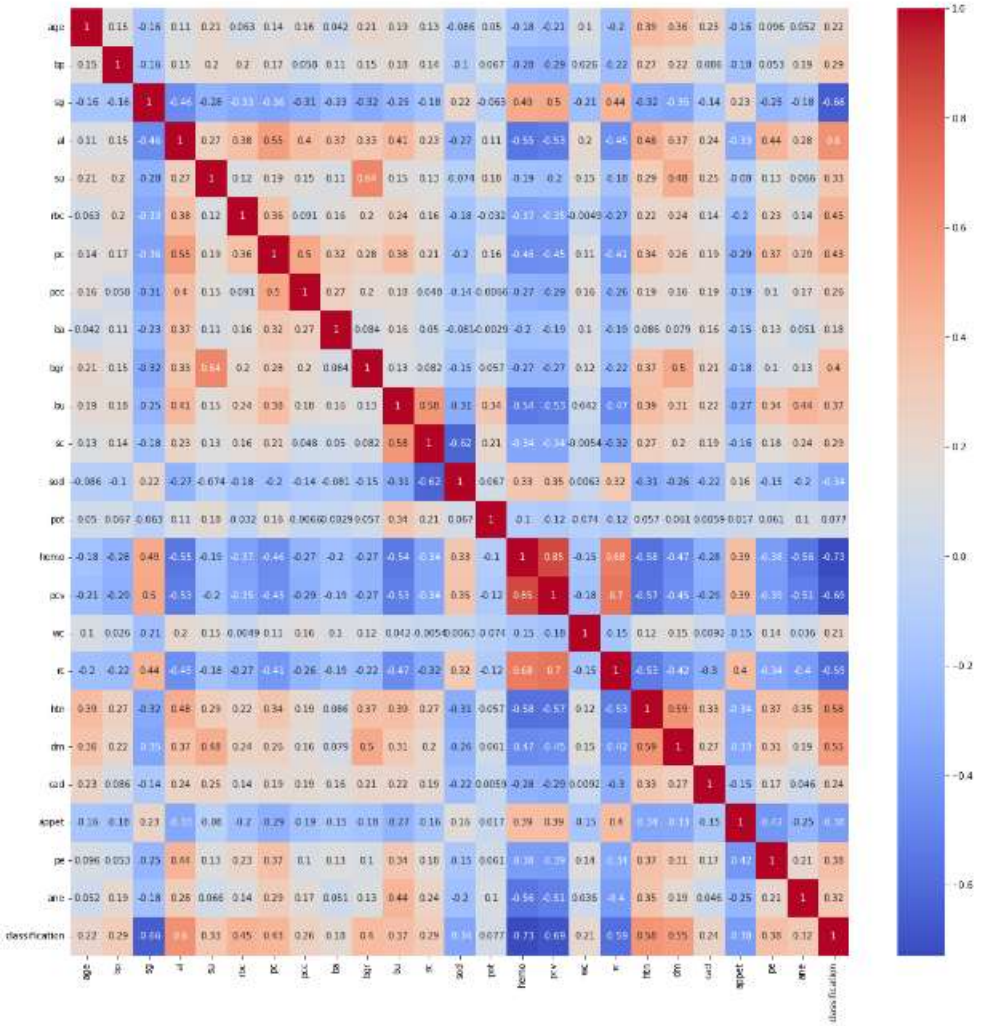
400 rows × 25 columns

In [30]:
```
data['classification'].value_counts()
```

Out[30]:
```
1.00    248
0.00    150
0.62      2
Name: classification, dtype: int64
```

In [31]:
```
plt.figure(figsize = (19,19))
sns.heatmap(data.corr(), annot = True, cmap = 'coolwarm')
```

Out[31]: <AxesSubplot:>



In [32]:
```
data.shape
```

Out[32]: (400, 25)

In [33]:
```
data.isnull().sum()
```

Out[33]:
```
age             0
bp              0
sg              0
al              0
su              0
rbc             0
pc              0
pcc             0
ba              0
bgr             0
bu              0
sc              0
sod             0
pot             0
hemo            0
pcv             0
wc              0
rc              0
htn             0
dm              0
cad             0
appet           0
pe              0
ane             0
classification  0
dtype: int64
```

LABEL ENCODING

In [35]:
```
from sklearn.linear_model import LogisticRegression
```

In [36]:
```
from sklearn.model_selection import train_test_split
```

In [37]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
```

SPLITTING DATASET INTO DEPENDENT AND INDEPENDENT VARIABLE

In [38]:
```
x = data.iloc[:,:-1]
y = data['classification']
```

In [39]:
```
lab_enc = preprocessing.LabelEncoder()
y = lab_enc.fit_transform(y)
```

SPLITTING THE DATASET INTO TRAINING AND TEST TEST

In [40]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.31, random_state =2029)
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
```

In [42]:
```
sc = StandardScaler()
```

MODEL BUILDING

In [43]:
```
logreg = LogisticRegression(solver='lbfgs',max_iter=10000)
```

In [44]:
```
logreg.fit(X_train,y_train)
```

Out[44]:
```
LogisticRegression(max_iter=10000)
```

TESTING THE MODEL

In [45]:
```
test_pred = logreg.predict(X_test)
train_pred = logreg.predict(X_train)
```

In [46]:
```
import sklearn.metrics as metrics
```

SAVE THE MODEL

In [49]:
```
filename = "Completed_model.joblib"
```

In [50]:
```
joblib.dump(logreg, filename)
```

Out[50]:
```
['Completed_model.joblib']
```

In [51]:
```
loaded_model = joblib.load(filename)
result = loaded_model.score(X_test, y_test)
print(result)
```

```
0.9758064516129032
```