

```

import pandas as pd #used for data manipulation

import numpy as np #used for numerical analysis

from collections import Counter as c # return counts of number of classess

import matplotlib.pyplot as plt #used for data visualization

import seaborn as sns #data visualization librory

import missingno as msno #finding missing values

from sklearn.metrics import accuracy_score,confusion_matrix#model performance

from sklearn.model_selection import train_test_split #splics data in random train and test array

from sklearn.preprocessing import LabelEncoder #encoding the levels of categorical fetures

from sklearn.linear_model import LogisticRegression #classification ML algorithm

import pickle #python object hierarchy is converted into a byte stream,

data = pd.read_csv(r"/content/kidney_disease.csv")# loading the csv data

data.head() #return you the first 5 rows values

```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44 7
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38 6
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31 7
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32 6
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35 7

5 rows × 26 columns



```
data.columns #return all the column names
```

```

Index(['id', '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')

```

```
data.shape
```

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

```

data.columns = ['age', 'blood_pressure', 'specific_gravity', 'albumin',
               'sugar', 'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
               'blood_glucose_randon', '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']#manyally giving the name off the columns

```

```
data.columns
```

```
Index(['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
      'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
      'blood_glucose_randon', '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'],
      dtype='object')
```

data.info()#info will give you a summary of dataset

```
<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_randon                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
```

data.isnull().any() #it will return ture if any columns is having null values

```
age                True
blood_pressure     True
specific_gravity   True
albumin            True
sugar              True
red_blood_cells    True
pus_cell           True
pus_cell_clumps    True
bacteria           True
blood_glucose_randon True
blood_urea         True
serum_creatinine   True
sodium             True
potassium          True
haemoglobin        True
packed_cell_volume True
white_blood_cell_count True
red_blood_cell_count True
hypertension       True
diabetes_mellitus   True
coronary_artery_disease True
appetite           True
peda_edema         True
aanemia            True
class              False
dtype: bool
```

#extracting numatic and categorical data

```
num_cols = [col for col in data.columns if data[col].dtype != 'object']
cat_cols = [col for col in data.columns if data[col].dtype == 'object']
```

```
num_cols
```

```
['age',
 'blood_pressure',
 'specific_gravity',
 'albumin',
 'sugar',
 'blood_glucose_randon',
 'blood_urea',
 'serum_creatinine',
 'sodium',
 'potassium',
 'haemoglobin']
```

```
#####check unique values in the categorical data
```

```
for col in cat_cols:
```

```
    print(f"{col} has {data[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
```

```
#####to handel the skewness in the data
```

```
def handel_outlier(col):
```

```
    data[col] =np.log1p(data[col])
```

```
    handel_outlier('blood_urea')
```

```
    handel_outlier('sodium')
```

```
handel_outlier('potassium')
handel_outlier('serum_creatinine')
handel_outlier('sugar')
```

```
##that explins why soe data still skeness although we to process on it, beacaus it stil has null data
data.isna().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  70
white_blood_cell_count  105
red_blood_cell_count  130
hypertension  2
diabetes_mellitus  2
coronary_artery_disease  2
appetite     1
peda_edema   1
aanemia      1
class        0
dtype: int64
```

```
#filling null values,we will use two methods, rndom sampling for higher null values and
# mean/mide sampling for lower null values
```

```
def random_value_imputation(feature):
    random_sample = data[feature].dropna().sample(data[feature].isna().sum())
    random_sample.index = data[data[feature].isnull()].index
    data.loc[data[feature].isnull(),feature] =random_sample
```

```
def impute_mode(feature):
    mode = data[feature].mode()[0]
    data[feature] =data[feature].fillna(mode)
```

```
###filling num columns null values uysing rando sampling method
```

```
for col in num_cols:
    random_value_imputation(col)
```

```
data[num_cols].isnull().sum()
```

```
age          0
blood_pressure  0
specific_gravity  0
albumin      0
sugar        0
blood_glucose_random  0
blood_urea    0
serum_creatinine  0
sodium       0
potassium    0
haemoglobin  0
dtype: int64
```

```
###label encoding for categorical data
from sklearn.preprocessing import LabelEncoder
```

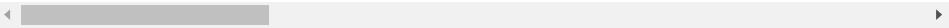
```
encode = LabelEncoder()
```

```
for col in cat_cols:
    data[col]=encode.fit_transform(data[col])
```

```
data.head()
```

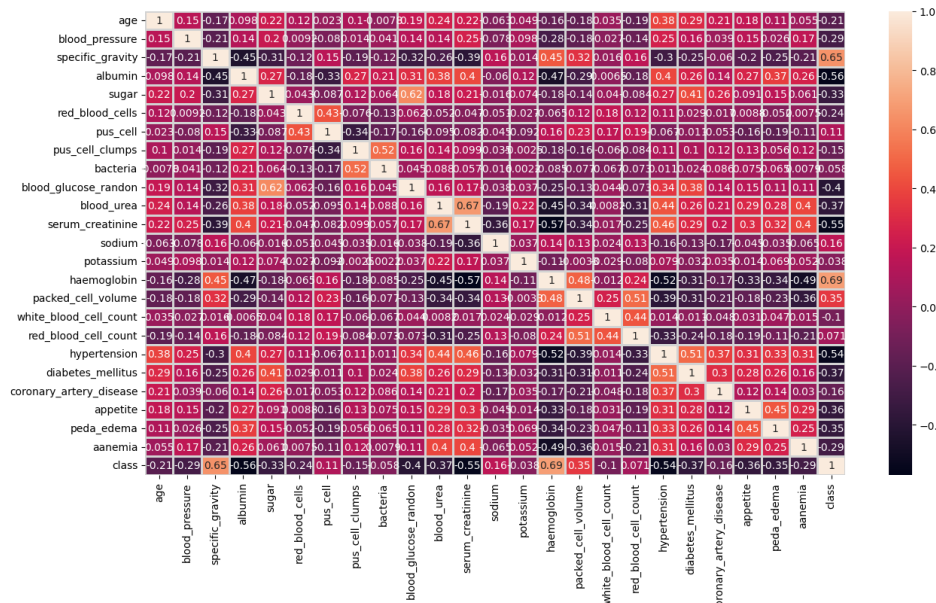
	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_
0	48.0	80.0	1.020	1.0	0.000000		2
1	7.0	50.0	1.020	4.0	0.000000		2
2	62.0	80.0	1.010	2.0	1.386294		1
3	48.0	70.0	1.005	4.0	0.000000		1
4	51.0	80.0	1.010	2.0	0.000000		1

5 rows × 25 columns



```
#heatmap of data
```

```
plt.figure(figsize = (15,8))
sns.heatmap(data.corr(),annot = True, linewidths =2, linecolor = 'lightgrey')
plt.show()
```



```
#looking at categorical columns
plt.figure(figsize =(20,15))
plotnumber = 1
```

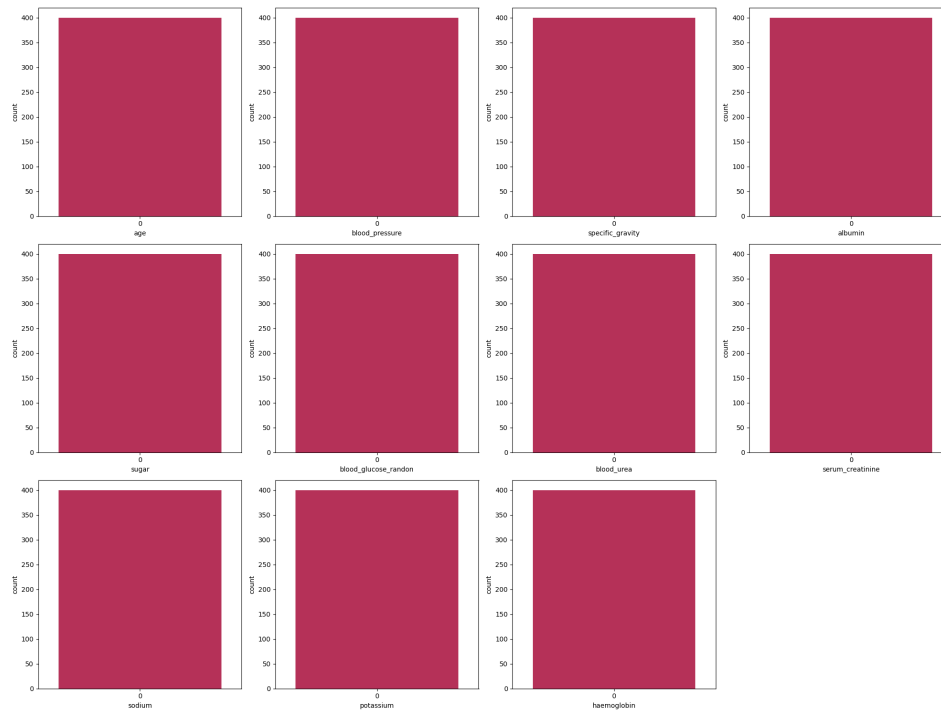
```

for column in num_cols:
    if plotnumber<=11:
        ax = plt.subplot(3, 4, plotnumber)
        sns.countplot(data[column],palette = 'rocket')
        plt.xlabel(column)

    plotnumber +=1

plt.tight_layout()
plt.show()

```



```
data[cat_cols].isnull().sum()
```

```

red_blood_cells      0
pus_cell              0
pus_cell_clumps      0
bacteria              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
peda_edema            0
aanemia               0
class                 0
dtype: int64

```

```
#checking for null values
```

```
data.isna().sum().sort_values(ascending = False)
```

```
age 0
potassium 0
aanemia 0
peda_edema 0
appetite 0
coronary_artery_disease 0
diabetes_mellitus 0
hypertension 0
red_blood_cell_count 0
white_blood_cell_count 0
packed_cell_volume 0
haemoglobin 0
sodium 0
blood_pressure 0
serum_creatinine 0
blood_urea 0
blood_glucose_randon 0
bacteria 0
pus_cell_clumps 0
pus_cell 0
red_blood_cells 0
sugar 0
albumin 0
specific_gravity 0
class 0
dtype: int64
```

```
#checking numerical features distribution
```

```
plt.figure(figsize = (20, 15))
plotnumber = 1
```

```
for column in num_cols:
    if plotnumber <=14:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(data[column])
        plt.xlabel(column)
```

```
    plotnumber += 1
```

```
plt.tight_layout()
plt.show()
```

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data[column])
<ipython-input-53-89eda4e8a1a5>:7: UserWarning:
```

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data[column])
<ipython-input-53-89eda4e8a1a5>:7: UserWarning:
```

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

