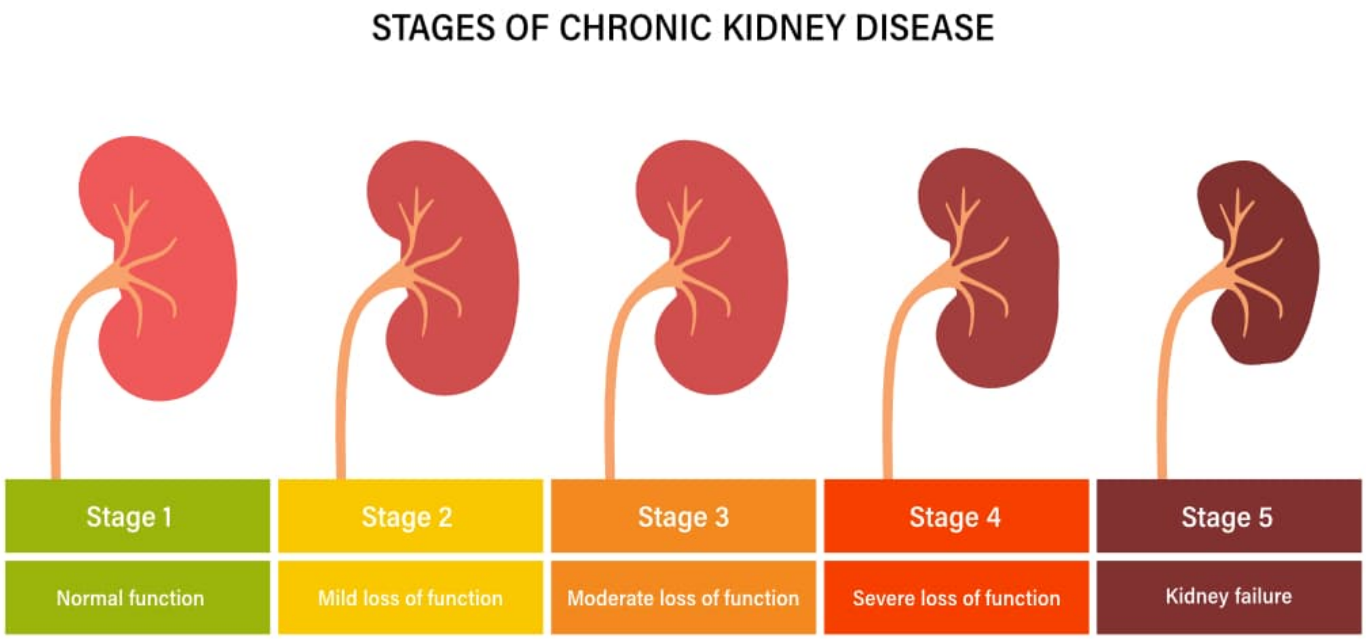


In [ ]:

```python
# import necessary libraries like numpy, pandas, pyplot and seaborn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```python
# reading dataset
chronic_df = pd.read_csv('/content/kidney_disease.csv')
chronic_df.head()
```

Out[ ]:

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

5 rows × 26 columns

Dataset Description

- age - age
- 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 - hemoglobin
- 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
- class - class
- ### %
- Number of Attributes: 24 + class = 25 (11 numeric ,14 nominal) %

Attribute Information :

- 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.Hemoglobin(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)
  `appet - (good,poor)`
- 23.Pedal Edema(nominal)
  `pe - (yes,no)`
- 24.Anemia(nominal)
  `ane - (yes,no)`
- 25.Class (nominal)
  `class - (ckd,notckd)`

In [ ]:

```
# checking info of columns and null values
```

```
<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
```

Data Cleaning

In [ ]:

```
# drop id column
chronic_df =
```

In [ ]:

```
# rename column names to make it more user-friendly

chronic_df.columns = ['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 're
              'pus_cell_clumps', 'bacteria', 'blood_glucose_random', 'blood_urea', 'serum_
              'potassium', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_count',
              'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite',
              'aanemia', 'class']
```

In [ ]:

```
chronic_df.head()
```

Out[ ]:

	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_cell	pus_cell_clumps	bacteria
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent

5 rows × 25 columns

According to the data description

- Cols(pcv, wc and rc) needs to convert back in numerical since it is object right now
- Cols(sg, al and su) should be nominal , convert from float to object

In [ ]:

```
# Categorical cols like specific_gravity, albumin and sugar which is float type right now
# converting back to nominal data type categorical
```

In [ ]:

```
# converting necessary columns like packed_cell_volume, white-blood_cell_count and red_blo
# currently it is in object type and converting back to numerical type
```

In [ ]:

```
# Extracting categorical and numerical columns

cat_cols =
num_cols =
```

In [ ]:

```
# by looping & looking at unique values in categorical columns
```

```
specific_gravity has [1.02 1.01 1.005 1.015 nan 1.025] values

albumin has [1.0 4.0 2.0 3.0 0.0 nan 5.0] values

sugar has [0.0 3.0 4.0 1.0 nan 2.0 5.0] values

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

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
```

In [ ]:

```
# replace incorrect values like '\tno', '\tyes', ' yes', '\tno', 'ckd\t', 'notckd'  in cat
```

In [ ]:

```
# Converting target col class into 0(chronic kidney) and 1(not a chronic kidney)



# coverting target col into numeric to check correlation
```

In [ ]:

```
# let's see the cols in numerical col list
```

Out[ ]:

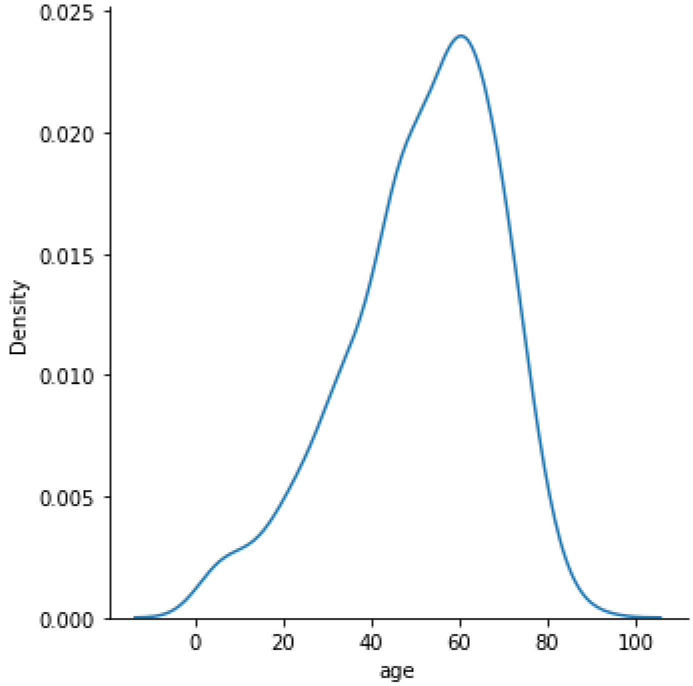
```
['age',
 'blood_pressure',
 'blood_glucose_random',
 'blood_urea',
 'serum_creatinine',
 'sodium',
 'potassium',
 'haemoglobin',
 'packed_cell_volume',
 'white_blood_cell_count',
 'red_blood_cell_count']
```

EDA

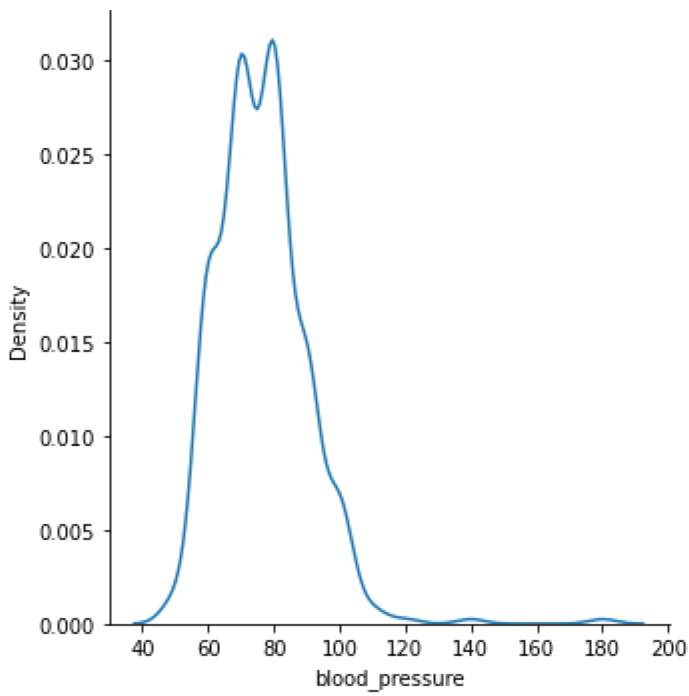
In [ ]:

```
# checking numerical features distribution


plt.figure(figsize=(20,12))

# looping over num cols and checking its distribution
```

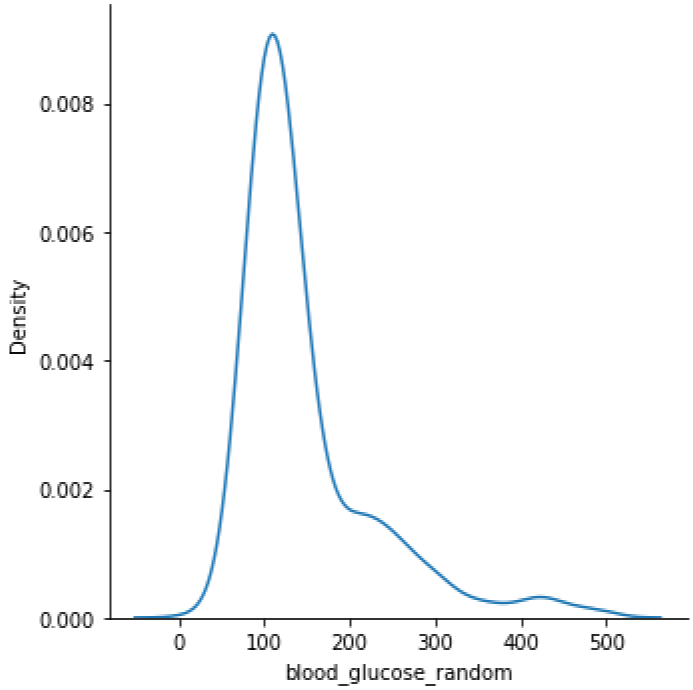
```
<Figure size 1440x864 with 0 Axes>
```



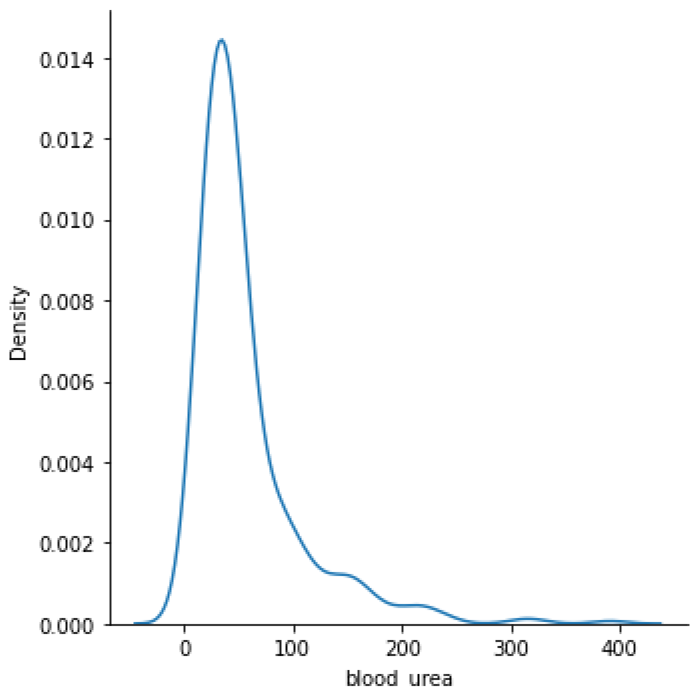
0.025
0.020
0.015
0.010
0.005
0.000
Density
0
20
40
60
80
100
age



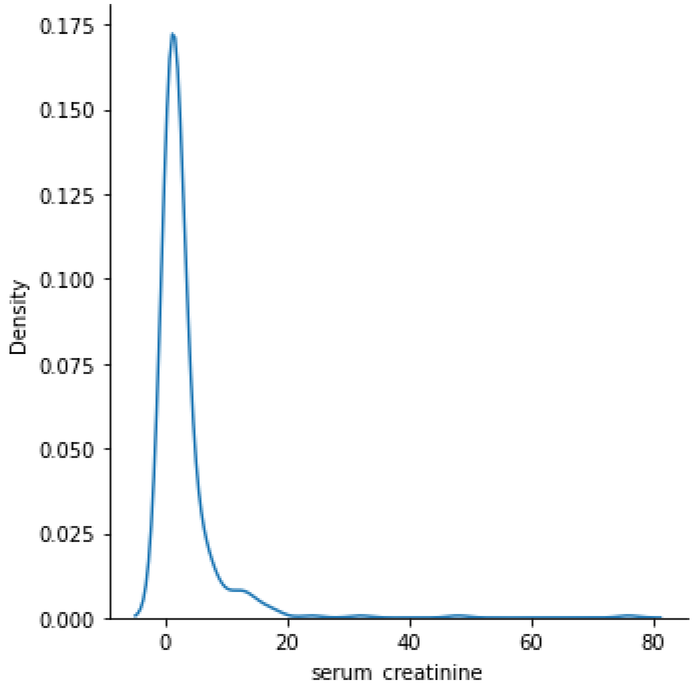
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
40
60
80
100
120
140
160
180
200
blood_pressure

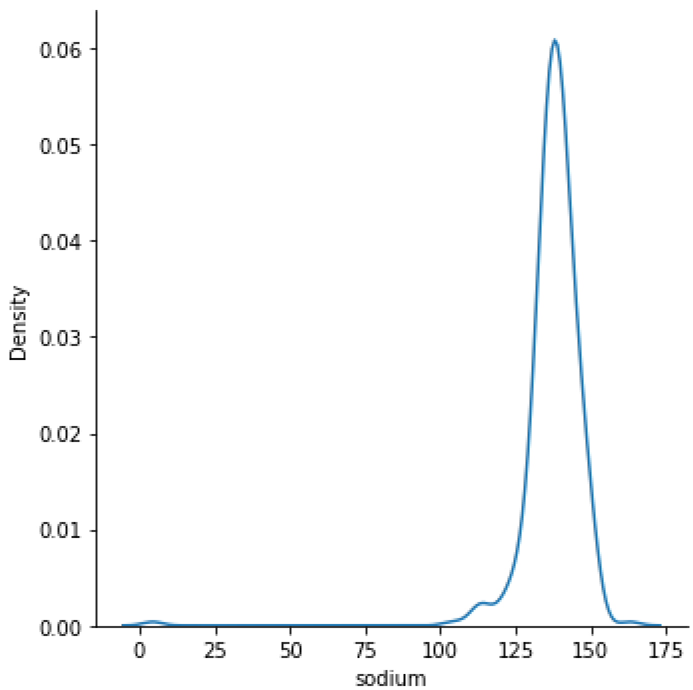


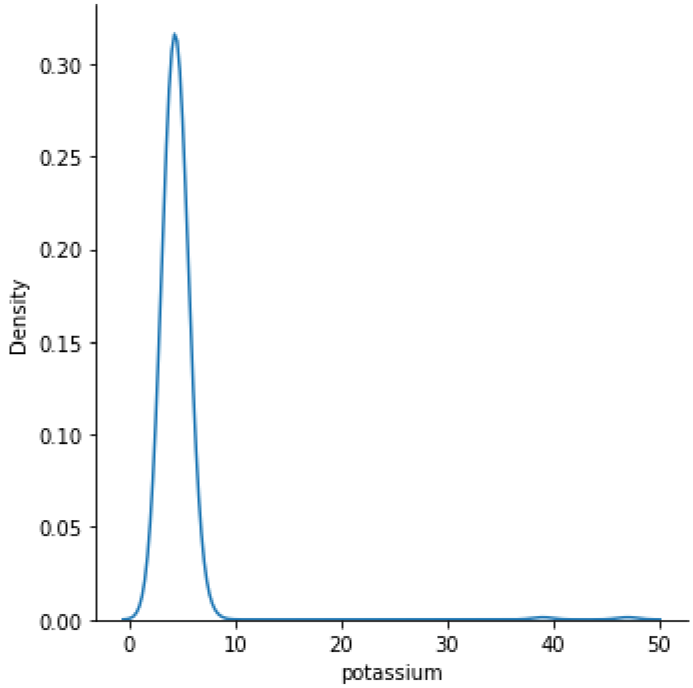
0.008
0.006
0.004
0.002
0.000
Density
0
100
200
300
400
500
blood_glucose_random

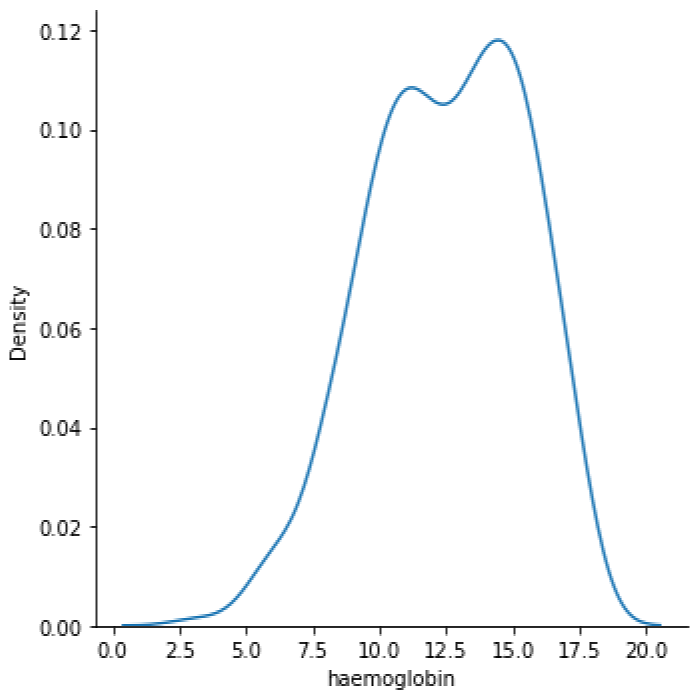


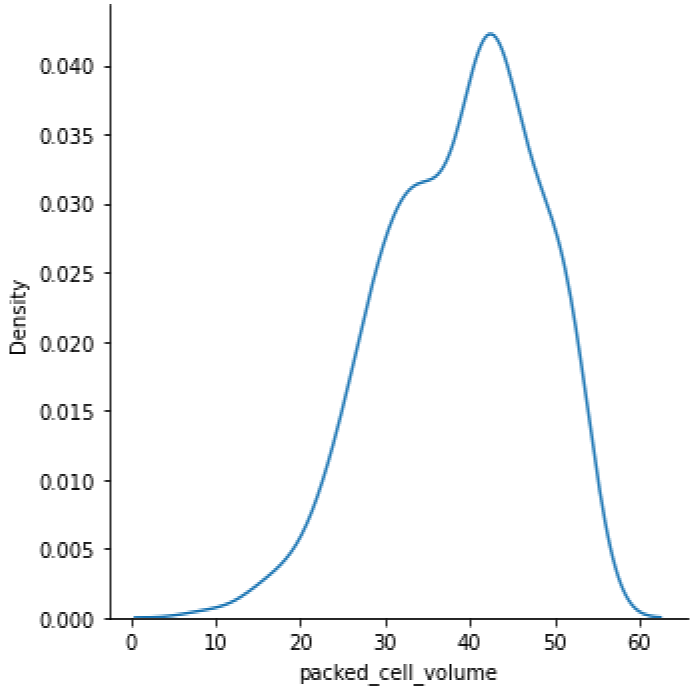
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
0
100
200
300
400
blood_urea


0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Density
0
20
40
60
80
serum_creatinine

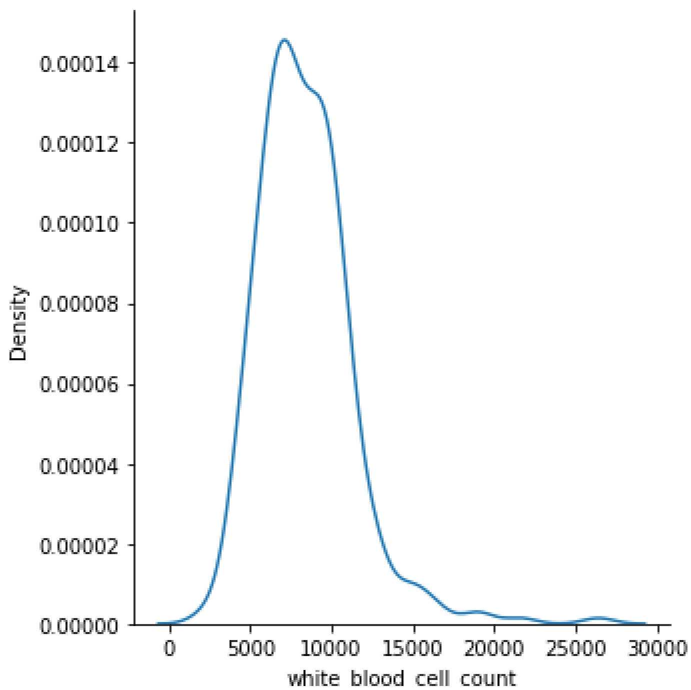

0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
0
25
50
75
100
125
150
175
sodium


0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0
10
20
30
40
50
potassium

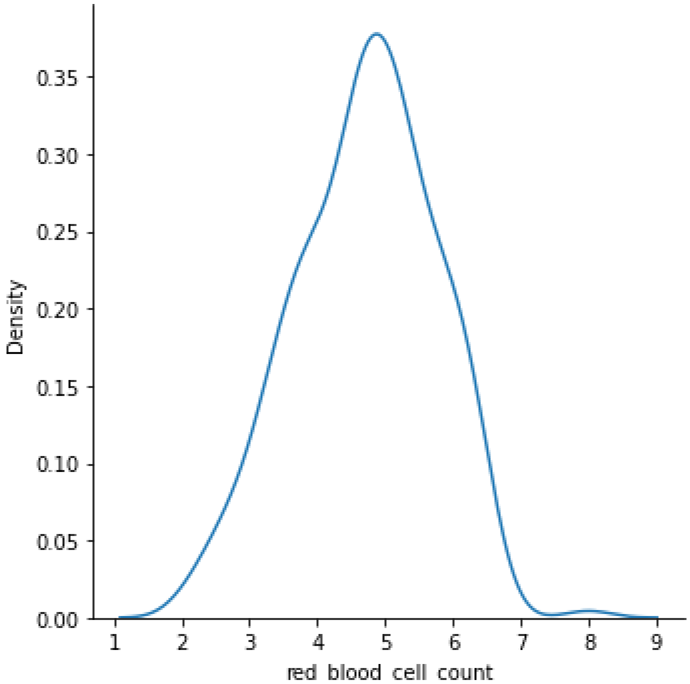

0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
haemoglobin



0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
0
10
20
30
40
50
60
packed_cell_volume



0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
Density
0
5000
10000
15000
20000
25000
30000
white_blood_cell_count



In [ ]:

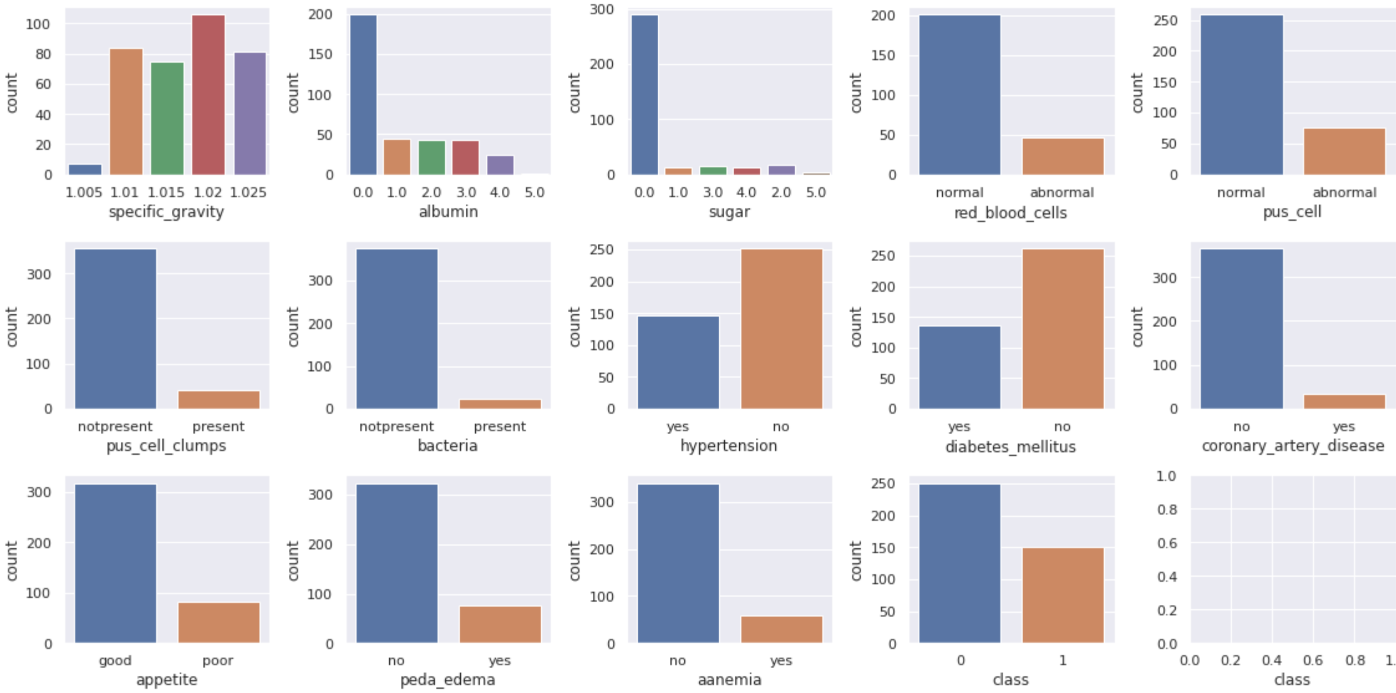
```
# let's see the cols in cat col list
```

Out[ ]:

```
['specific_gravity',
 'albumin',
 'sugar',
 'red_blood_cells',
 'pus_cell',
 'pus_cell_clumps',
 'bacteria',
 'hypertension',
 'diabetes_mellitus',
 'coronary_artery_disease',
 'appetite',
 'peda_edema',
 'aanemia',
 'class']
```

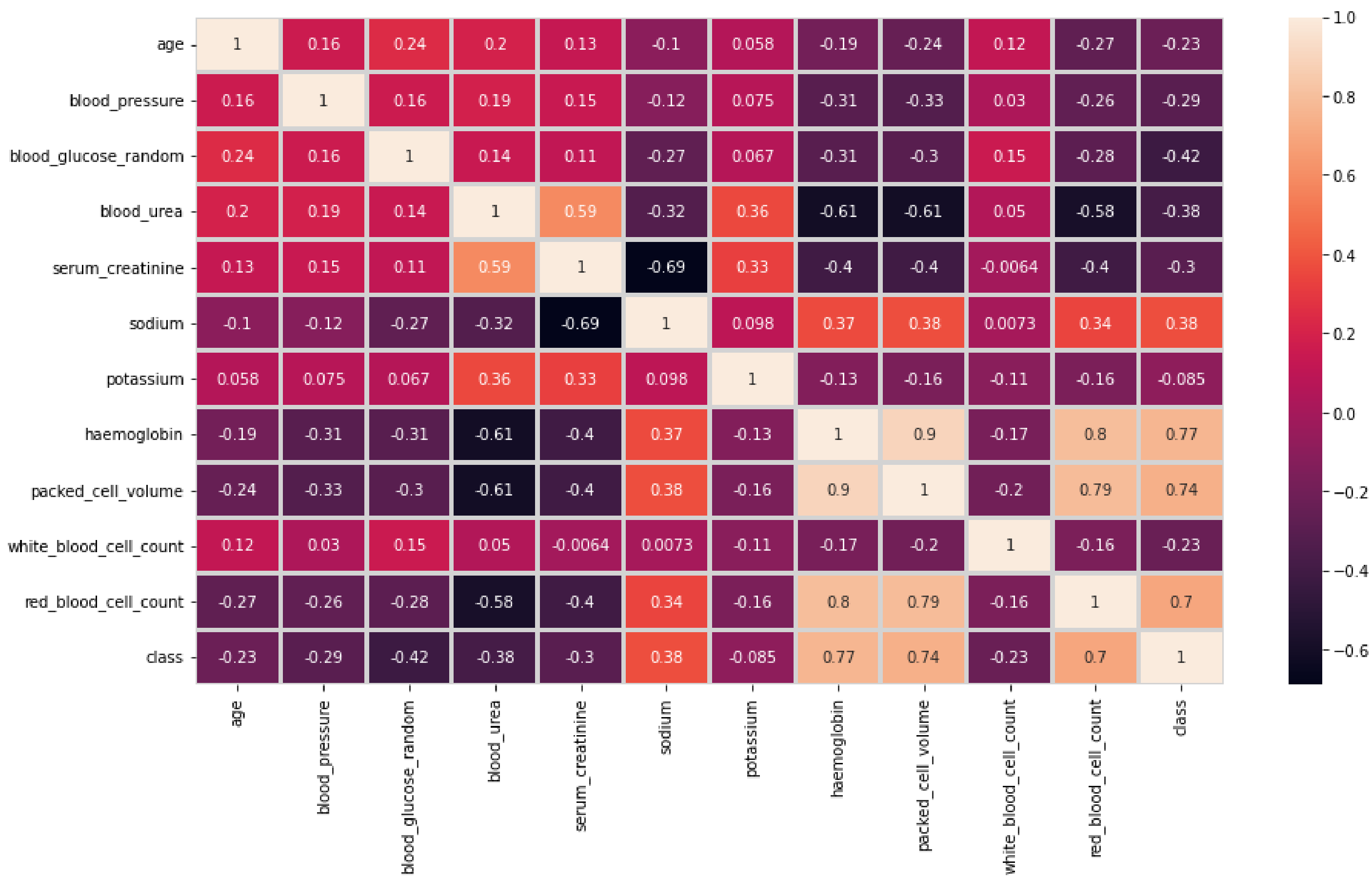
In [ ]:

```
# checking cat features distribution

# create the figure and axes

fig, axes = plt.subplots(3, 5, figsize=(16,8))


axes = axes.ravel()  # flattening the array makes indexing easier

# loop over cat cols and plot countplot
```



In [ ]:

```python
# correlated heatmap of data

plt.figure(figsize = (15, 8))
```



In [ ]:

```python
# let's check count of null values in whole df
```

Out[ ]:

```
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
appetite                     1
peda_edema                   1
aanemia                      1
class                        0
dtype: int64
```

In [ ]:

```
# let's check count of null values in num_cols
```

Out[ ]:

```
age                         9
blood_pressure             12
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
dtype: int64
```

In [ ]:

```
# let's check count of null values in cat cols
```

Out[ ]:

```
specific_gravity            47
albumin                     46
sugar                       49
red_blood_cells            152
pus_cell                    65
pus_cell_clumps              4
bacteria                     4
hypertension                 2
diabetes_mellitus            2
coronary_artery_disease      2
appetite                     1
peda_edema                   1
aanemia                      1
class                        0
dtype: int64
```

Missing Value Treatment

In [ ]:

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

# creating func for imputing random values
def random_value_imputation(feature):
```

```
    random_sample =
    random_sample.index =
    chronic_df.loc[chronic_df[feature].isnull(), feature] =

# creating func for imputing most common value(modal value)
def impute_mode(feature):
    mode =
    chronic_df[feature] =
```

In [ ]:
```
# filling num_cols null values using random sampling method
```

In [ ]:
```
# let's check count of null values in num_cols again
```

Out[ ]:
```
age                        0
blood_pressure             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
dtype: int64
```

In [ ]:
```
# filling "red_blood_cells" and "pus_cell" using random sampling method and rest of cat_co
```

In [ ]:
```
# let's check count of null values in cat_cols again
```

Out[ ]:
```
specific_gravity           0
albumin                    0
sugar                      0
red_blood_cells            0
pus_cell                   0
pus_cell_clumps            0
bacteria                   0
hypertension               0
diabetes_mellitus          0
coronary_artery_disease    0
appetite                   0
peda_edema                 0
aanemia                    0
class                      0
dtype: int64
```

In [ ]:
```
# check unique values in each cat col by looping over cat cols
```

```
specific_gravity has 5 categories

albumin has 6 categories

sugar has 6 categories

red_blood_cells has 2 categories

pus_cell has 2 categories

pus_cell_clumps has 2 categories
```

```
bacteria has 2 categories

hypertension has 2 categories

diabetes_mellitus has 2 categories

coronary_artery_disease has 2 categories

appetite has 2 categories

peda_edema has 2 categories

aanemia has 2 categories

class has 2 categories
```

In [ ]:

```python
# using labelencoder and applying on cat cols
from sklearn.preprocessing import LabelEncoder

le =

for col in cat_cols[3:]:
    chronic_df[col] =
```

In [ ]:

```python
# check chronic df after transforming cat cols
```

Out[ ]:

	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_cell	pus_cell_clumps	bacteria	b
0	48.0	80.0	1.020	1.0	0.0	1	1	0	0	
1	7.0	50.0	1.020	4.0	0.0	1	1	0	0	
2	62.0	80.0	1.010	2.0	3.0	1	1	0	0	
3	48.0	70.0	1.005	4.0	0.0	1	0	1	0	
4	51.0	80.0	1.010	2.0	0.0	1	1	0	0	

5 rows × 25 columns

In [ ]:

```python
# Split data into features and target variables (X and y)
X =
y =
```

In [ ]:

```python
# splitting data intp training and test set, so import train_test_split

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
```

Model Building

In [ ]:

```python
# import KNeighborsClassifier, accuracy_score, confusion_matrix, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

knn =
```

```python
# accuracy score, confusion matrix and classification report of knn

knn_acc =
```

```
Training Accuracy of KNN is 0.7857142857142857
Test Accuracy of KNN is 0.6333333333333333

Confusion Matrix :-
[[45 27]
 [17 31]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.73      0.62      0.67        72
           1       0.53      0.65      0.58        48

    accuracy                           0.63       120
   macro avg       0.63      0.64      0.63       120
weighted avg       0.65      0.63      0.64       120
```

In [ ]:

```python
# import DecisionTreeClassifer
from sklearn.tree import DecisionTreeClassifier

dtc =




# accuracy score, confusion matrix and classification report of decision tree

dtc_acc =
```

```
Training Accuracy of Decision Tree Classifier is 1.0
Test Accuracy of Decision Tree Classifier is 0.9833333333333333

Confusion Matrix :-
[[70  2]
 [ 0 48]]

Classification Report :-
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        72
           1       0.96      1.00      0.98        48

    accuracy                           0.98       120
   macro avg       0.98      0.99      0.98       120
weighted avg       0.98      0.98      0.98       120
```

In [ ]:

```python
# hyper parameter tuning of decision tree , import GridSearchCV

from sklearn.model_selection import GridSearchCV

"""
Use this param
 {
```

```
    'criterion' : ['gini', 'entropy'],
    'max_depth' : [3, 5, 7, 10],
    'splitter' : ['best', 'random'],
    'min_samples_leaf' : [1, 2, 3, 5, 7],
    'min_samples_split' : [1, 2, 3, 5, 7],
    'max_features' : ['auto', 'sqrt', 'log2']
}

"""

grid_param = 


# Apply gridsearchcv with cv = 5, n_jobs = -1, verbose = 1
grid_search_dtc = 
```

```
Fitting 5 folds for each of 1200 candidates, totalling 6000 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:372: FitFail
edWarning:
1200 fits failed out of a total of 6000.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='rais
e'.

Below are more details about the failures:
--------------------------------------------------------------------------------
1200 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py", li
ne 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line 942, in fit
    X_idx_sorted=X_idx_sorted,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line 254, in fit
    % self.min_samples_split
ValueError: min_samples_split must be an integer greater than 1 or a float in (0.0, 1.0];
got the integer 1

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarnin
g: One or more of the test scores are non-finite: [       nan        nan 0.94285714 ... 0.
85       0.93928571 0.94285714]
  category=UserWarning,
```

Out[ ]:

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 7, 10],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 3, 5, 7],
                         'min_samples_split': [1, 2, 3, 5, 7],
                         'splitter': ['best', 'random']},
             verbose=1)
```

In [ ]:

```
# print best parameters and best score in grid search dtc
```

```
{'criterion': 'gini', 'max_depth': 7, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_
samples_split': 2, 'splitter': 'best'}
0.9857142857142858
```

In [ ]:

```
# storing best estimator

dtc = 

# accuracy score, confusion matrix and classification report of decision tree
```

```python
dtc_acc = 
```

```
Training Accuracy of Decision Tree Classifier is 0.9821428571428571
Test Accuracy of Decision Tree Classifier is 0.9833333333333333

Confusion Matrix :-
[[72  0]
 [ 2 46]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        72
           1       1.00      0.96      0.98        48

    accuracy                           0.98       120
   macro avg       0.99      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

In [ ]:

```python
# import RandomForestClassifier

from sklearn.ensemble import RandomForestClassifier

rd_clf = 



# accuracy score, confusion matrix and classification report of random forest

rd_clf_acc = 
```

```
Training Accuracy of Random Forest Classifier is 0.9964285714285714
Test Accuracy of Random Forest Classifier is 0.9666666666666667

Confusion Matrix :-
[[71  1]
 [ 3 45]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        72
           1       0.98      0.94      0.96        48

    accuracy                           0.97       120
   macro avg       0.97      0.96      0.97       120
weighted avg       0.97      0.97      0.97       120
```

In [ ]:

```python
# import AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier

ada = 


# accuracy score, confusion matrix and classification report of ada boost

ada_acc = 
```

```
Training Accuracy of Ada Boost Classifier is 1.0
Test Accuracy of Ada Boost Classifier is 0.975
```

```
Confusion Matrix :-
[[71  1]
 [ 2 46]]

Classification Report :-
               precision    recall  f1-score   support

           0       0.97      0.99      0.98        72
           1       0.98      0.96      0.97        48

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

In [ ]:

```python
# import GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingClassifier

gb =


# accuracy score, confusion matrix and classification report of gradient boosting classifi

gb_acc =
```

```
Training Accuracy of Gradient Boosting Classifier is 1.0
Test Accuracy of Gradient Boosting Classifier is 0.9833333333333333

Confusion Matrix :-
[[71  1]
 [ 1 47]]

Classification Report :-
               precision    recall  f1-score   support

           0       0.99      0.99      0.99        72
           1       0.98      0.98      0.98        48

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

In [ ]:

```python
# using max_depth = 4, subsample = 0.90, max_features = 0.75, n_estimators = 200
sgb =




# accuracy score, confusion matrix and classification report of stochastic gradient boosti

sgb_acc =
```

```
Training Accuracy of Stochastic Gradient Boosting is 1.0
Test Accuracy of Stochastic Gradient Boosting is 0.9833333333333333

Confusion Matrix :-
[[71  1]
 [ 1 47]]

Classification Report :-
               precision    recall  f1-score   support

           0       0.99      0.99      0.99        72
```

```
           1       0.98      0.98      0.98        48

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

In [ ]:

```python
# import XGBClassifier
from xgboost import XGBClassifier

xgb =


# accuracy score, confusion matrix and classification report of xgboost

xgb_acc =
```

```
Training Accuracy of XgBoost is 1.0
Test Accuracy of XgBoost is 0.9916666666666667

Confusion Matrix :-
[[71  1]
 [ 0 48]]

Classification Report :-
               precision    recall  f1-score   support

           0       1.00      0.99      0.99        72
           1       0.98      1.00      0.99        48

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

In [ ]:

```python
# pip install catboost
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publi
c/simple/
Collecting catboost
  Downloading catboost-1.1.1-cp37-none-manylinux1_x86_64.whl (76.6 MB)
     |████████████████████████████████| 76.6 MB 91 kB/s
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (fr
om catboost) (1.21.6)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from ca
tboost) (0.10.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from catb
oost) (5.5.0)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages (f
rom catboost) (1.3.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from catbo
ost) (1.7.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from
catboost) (3.2.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from catboos
t) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (fro
m pandas>=0.24.0->catboost) (2022.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-pac
kages (from pandas>=0.24.0->catboost) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (fro
m matplotlib->catboost) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/
python3.7/dist-packages (from matplotlib->catboost) (3.0.9)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib->catboost) (1.4.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages
(from kiwisolver>=1.0.1->matplotlib->catboost) (4.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages
(from plotly->catboost) (8.1.0)
Installing collected packages: catboost
Successfully installed catboost-1.1.1
```

In [ ]:

```python
# import CatBoostClassifier
from catboost import CatBoostClassifier

cat = 


# accuracy score, confusion matrix and classification report of cat boost

cat_acc = 
```

```
Learning rate set to 0.408198
0:	learn: 0.3355370	total: 56.2ms	remaining: 505ms
1:	learn: 0.1668871	total: 62.7ms	remaining: 251ms
2:	learn: 0.0942740	total: 70.3ms	remaining: 164ms
3:	learn: 0.0714509	total: 79.4ms	remaining: 119ms
4:	learn: 0.0556378	total: 88.1ms	remaining: 88.1ms
5:	learn: 0.0448667	total: 93.5ms	remaining: 62.4ms
6:	learn: 0.0314548	total: 101ms	remaining: 43.1ms
7:	learn: 0.0266063	total: 108ms	remaining: 26.9ms
8:	learn: 0.0209733	total: 114ms	remaining: 12.6ms
9:	learn: 0.0193500	total: 125ms	remaining: 0us
Training Accuracy of Cat Boost Classifier is 1.0
Test Accuracy of Cat Boost Classifier is 0.9833333333333333

Confusion Matrix :-
[[71  1]
 [ 1 47]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        72
           1       0.98      0.98      0.98        48

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

In [ ]:

```python
# import ExtraTreesClassifier

from sklearn.ensemble import ExtraTreesClassifier

etc = 




# accuracy score, confusion matrix and classification report of extra trees classifier

etc_acc = 
```

```
Training Accuracy of Extra Trees Classifier is 1.0
Test Accuracy of Extra Trees Classifier is 1.0
```

```
Confusion Matrix :-
[[72  0]
 [ 0 48]]

Classification Report :-
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        72
           1       1.00      1.00      1.00        48

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

In [ ]:

```
# import LGBMClassifier
from lightgbm import LGBMClassifier

lgbm =




# accuracy score, confusion matrix and classification report of lgbm classifier

lgbm_acc =
```

```
Training Accuracy of LGBM Classifier is 1.0
Test Accuracy of LGBM Classifier is 0.9916666666666667

[[71  1]
 [ 0 48]]

              precision    recall  f1-score   support

           0       1.00      0.99      0.99        72
           1       0.98      1.00      0.99        48

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

In [ ]:

```
# comparing all models accuracy by creating a df
models =
```

Out[ ]:

	Model	Score
8	Extra Trees Classifier	1.000000
6	XgBoost	0.991667
1	Decision Tree Classifier	0.983333
4	Gradient Boosting Classifier	0.983333
5	Stochastic Gradient Boosting	0.983333
7	Cat Boost	0.983333
3	Ada Boost Classifier	0.975000
2	Random Forest Classifier	0.966667
0	KNN	0.633333

```
In [ ]:
```