1. Imports modules and necessary packages

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import os
import feather
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.feature_selection import SelectKBest
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import chi2
from utility_func.utility import *
from pandas_summary import DataFrameSummary
from sklearn import metrics as met
import joblib
```

In [2]:
```python
np.random.seed(42)
np.set_printoptions(suppress=True)
warnings.filterwarnings('ignore')
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 20
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

In [3]:
```python
df = pd.read_csv('input/data/raw_data.csv') # Reading the Dataset
```

2. Data Analysis

In [4]:
```python
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 [5]:
```python
# rename the catagory column
df.rename(columns={'classification' : 'class'}, inplace=True)
```

In [6]:
```python
df.columns
```

Out[6]:
```
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', 'class'],
      dtype='object')
```

In [7]:
```python
df.drop('id', axis=1, inplace=True)
```

In [8]:
```python
df.columns
```

Out[8]:
```
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', 'class'],
      dtype='object')
```

2.1 Data Preprocessing :Finding and Removing Anomalies

In [9]:
```python
# Function to find the categorical and numerical features
def extract_cat_num(df):
    cat_col=[col for col in df.columns if df[col].dtype=='O']
    num_col=[col for col in df.columns if df[col].dtype!='O']
    return cat_col,num_col
```

In [10]:
```python
cat_col,num_col=extract_cat_num(df)
cat_col
```

Out[10]:
```
['rbc',
 'pc',
 'pcc',
 'ba',
 'pcv',
 'wc',
 'rc',
 'htn',
 'dm',
 'cad',
 'appet',
 'pe',
 'ane',
 'class']
```

In [11]:
```python
num_col
```

Out[11]:
```
['age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hemo']
```

In [12]:
```python
# dirtiness in data

for col in cat_col:
    print('{} has {} values'.format(col,df[col].unique()))
    print("\n")
```

```
rbc has [nan 'normal' 'abnormal'] values


pc has ['normal' 'abnormal' nan] values


pcc has ['notpresent' 'present' nan] values


ba has ['notpresent' 'present' nan] values


pcv 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


wc 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


rc 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


htn has ['yes' 'no' nan] values


dm has ['yes' 'no' ' yes' '\tno' '\tyes' nan] values


cad has ['no' 'yes' '\tno' nan] values


appet has ['good' 'poor' nan] values


pe has ['no' 'yes' nan] values


ane has ['no' 'yes' nan] values


class has ['ckd' 'ckd\t' 'notckd'] values
```

In [13]:
```python
# Convert the object string  features into float
for col in ['pcv','wc','rc']:
    df[col] = df[col].str.extract('(\d+\.\d+|\d+)').astype(float)
```

In [14]:
```python
# Remove the tab in the categorical variables
df['cad'].replace(to_replace={'\tno':'no'}, inplace=True)
df['dm'].replace(to_replace={'\tno':'no','\tyes':'yes',' yes':'yes'}, inplace=True)
df['class'].replace(to_replace={'ckd\t':'ckd'},inplace=True)
```

2.2 Save the preprocessed data

In [15]:
```python
os.makedirs('./input/processed/', exist_ok=True)
df.to_feather('./input/processed/ckd-processed')
```

In [16]:
```python
df = feather.read_dataframe('input/processed/ckd-processed')
df.head()
```

Out[16]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	class
0	48.0	80.0	1.020	1.0	0.0	None	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	None	normal	notpresent	notpresent	NaN	18.0	0.8	NaN	NaN	11.3	38.0	6000.0	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	53.0	1.8	NaN	NaN	9.6	31.0	7500.0	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	56.0	3.8	111.0	2.5	11.2	32.0	6700.0	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	26.0	1.4	NaN	NaN	11.6	35.0	7300.0	4.6	no	no	no	good	no	no	ckd

In [17]:
```python
for cols in ['sg', 'al', 'su']:
    df[cols] = df[cols].astype('category')
```

In [18]:
```python
#Change columns of strings in dataframe to a column of categorical values
train_cats(df)
df.info()
```

```
<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   bp      388 non-null    float64
 2   sg      353 non-null    category
 3   al      354 non-null    category
 4   su      351 non-null    category
 5   rbc     248 non-null    category
 6   pc      335 non-null    category
 7   pcc     396 non-null    category
 8   ba      396 non-null    category
 9   bgr     356 non-null    float64
 10  bu      381 non-null    float64
 11  sc      383 non-null    float64
 12  sod     313 non-null    float64
 13  pot     312 non-null    float64
 14  hemo    348 non-null    float64
 15  pcv     329 non-null    float64
 16  wc      294 non-null    float64
 17  rc      269 non-null    float64
 18  htn     398 non-null    category
 19  dm      398 non-null    category
 20  cad     398 non-null    category
 21  appet   399 non-null    category
 22  pe      399 non-null    category
 23  ane     399 non-null    category
 24  class   400 non-null    category
dtypes: category(14), float64(11)
memory usage: 41.6 KB
```

In [19]:
```python
df['class'].cat.categories
```

Out[19]:
```
Index(['ckd', 'notckd'], dtype='object')
```

In [20]:
```python
# collect the categorical varibales in dictionary
for cols in df.columns:
    if df[cols].dtype.name == 'category':
        print(cols, dict(enumerate(df[cols].cat.categories)) )
```

```
sg {0: 1.005, 1: 1.01, 2: 1.015, 3: 1.02, 4: 1.025}
al {0: 0.0, 1: 1.0, 2: 2.0, 3: 3.0, 4: 4.0, 5: 5.0}
su {0: 0.0, 1: 1.0, 2: 2.0, 3: 3.0, 4: 4.0, 5: 5.0}
rbc {0: 'abnormal', 1: 'normal'}
pc {0: 'abnormal', 1: 'normal'}
pcc {0: 'notpresent', 1: 'present'}
ba {0: 'notpresent', 1: 'present'}
htn {0: 'no', 1: 'yes'}
dm {0: 'no', 1: 'yes'}
cad {0: 'no', 1: 'yes'}
appet {0: 'good', 1: 'poor'}
pe {0: 'no', 1: 'yes'}
ane {0: 'no', 1: 'yes'}
class {0: 'ckd', 1: 'notckd'}
```

In [21]:
```python
# convert the appet and class categories for 1 for good/positive and 0 for poor/ negative to make constitent
df['appet'].cat.set_categories(['poor', 'good'], ordered=True, inplace=True)
df['class'].cat.set_categories(['notckd', 'ckd'], ordered=True, inplace=True)
```

In [22]:
```python
for cols in df.columns:
    if df[cols].dtype.name == 'category':
        print(cols, dict(enumerate(df[cols].cat.categories)) )
```

```
sg {0: 1.005, 1: 1.01, 2: 1.015, 3: 1.02, 4: 1.025}
al {0: 0.0, 1: 1.0, 2: 2.0, 3: 3.0, 4: 4.0, 5: 5.0}
su {0: 0.0, 1: 1.0, 2: 2.0, 3: 3.0, 4: 4.0, 5: 5.0}
rbc {0: 'abnormal', 1: 'normal'}
pc {0: 'abnormal', 1: 'normal'}
pcc {0: 'notpresent', 1: 'present'}
ba {0: 'notpresent', 1: 'present'}
htn {0: 'no', 1: 'yes'}
dm {0: 'no', 1: 'yes'}
cad {0: 'no', 1: 'yes'}
appet {0: 'poor', 1: 'good'}
pe {0: 'no', 1: 'yes'}
ane {0: 'no', 1: 'yes'}
class {0: 'notckd', 1: 'ckd'}
```

In [23]:
```python
# Save the semi processed data
df.to_feather('./input/processed/ckd_semi_processed')
```

3. Creating Feature Variable and Target Variable

In [24]:
```python
df = feather.read_dataframe('input/processed/ckd_semi_processed')
df.head()
```

Out[24]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	class
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4	44.0	7800.0	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	18.0	0.8	NaN	NaN	11.3	38.0	6000.0	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	53.0	1.8	NaN	NaN	9.6	31.0	7500.0	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	56.0	3.8	111.0	2.5	11.2	32.0	6700.0	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	26.0	1.4	NaN	NaN	11.6	35.0	7300.0	4.6	no	no	no	good	no	no	ckd

In [25]:
```python
for cols in df.columns:
    if df[cols].dtype.name == 'category':
        print(cols, dict(enumerate(df[cols].cat.categories)) )
```

```
sg {0: 1.005, 1: 1.01, 2: 1.015, 3: 1.02, 4: 1.025}
al {0: 0.0, 1: 1.0, 2: 2.0, 3: 3.0, 4: 4.0, 5: 5.0}
su {0: 0.0, 1: 1.0, 2: 2.0, 3: 3.0, 4: 4.0, 5: 5.0}
rbc {0: 'abnormal', 1: 'normal'}
pc {0: 'abnormal', 1: 'normal'}
pcc {0: 'notpresent', 1: 'present'}
ba {0: 'notpresent', 1: 'present'}
htn {0: 'no', 1: 'yes'}
dm {0: 'no', 1: 'yes'}
cad {0: 'no', 1: 'yes'}
appet {0: 'poor', 1: 'good'}
pe {0: 'no', 1: 'yes'}
ane {0: 'no', 1: 'yes'}
class {0: 'notckd', 1: 'ckd'}
```

In [26]:
```python
# splits off the response variable, and replaced NAN by the median value of the column.
features, response, nas = proc_df(df, 'class')
```

In [27]:
```python
features.shape, response.shape
```

Out[27]:
```
((400, 35), (400,))
```

In [28]:
```python
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 35 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   age      400 non-null    float64
 1   bp       400 non-null    float64
 2   sg       400 non-null    int8
 3   al       400 non-null    int8
 4   su       400 non-null    int8
 5   rbc      400 non-null    int8
 6   pc       400 non-null    int8
 7   pcc      400 non-null    int8
 8   ba       400 non-null    int8
 9   bgr      400 non-null    float64
 10  bu       400 non-null    float64
 11  sc       400 non-null    float64
 12  sod      400 non-null    float64
 13  pot      400 non-null    float64
 14  hemo     400 non-null    float64
 15  pcv      400 non-null    float64
 16  wc       400 non-null    float64
 17  rc       400 non-null    float64
 18  htn      400 non-null    int8
 19  dm       400 non-null    int8
 20  cad      400 non-null    int8
 21  appet    400 non-null    int8
 22  pe       400 non-null    int8
 23  ane      400 non-null    int8
 24  age_na   400 non-null    bool
 25  bp_na    400 non-null    bool
 26  bgr_na   400 non-null    bool
 27  bu_na    400 non-null    bool
 28  sc_na    400 non-null    bool
 29  sod_na   400 non-null    bool
 30  pot_na   400 non-null    bool
 31  hemo_na  400 non-null    bool
 32  pcv_na   400 non-null    bool
 33  wc_na    400 non-null    bool
 34  rc_na    400 non-null    bool
dtypes: bool(11), float64(11), int8(13)
memory usage: 43.9 KB
```

In [29]:
```python
nas
```

Out[29]:
```
{'age': 55.0,
 'bp': 80.0,
 'bgr': 121.0,
 'bu': 42.0,
 'sc': 1.3,
 'sod': 138.0,
 'pot': 4.4,
 'hemo': 12.649999999999999,
 'pcv': 40.0,
 'wc': 8000.0,
 'rc': 4.8}
```

In [30]:
```python
cols = features.columns
cols
```

Out[30]:
```
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', 'age_na', 'bp_na', 'bgr_na', 'bu_na', 'sc_na',
       'sod_na', 'pot_na', 'hemo_na', 'pcv_na', 'wc_na', 'rc_na'],
      dtype='object')
```

3.1 Feature Scaling

In [31]:
```python
scaler = MinMaxScaler()
```

In [32]:
```python
x_features = scaler.fit_transform(features)
```

In [33]:
```python
x_features_scaled = pd.DataFrame(x_features, columns=cols)
x_features_scaled.head()
```

Out[33]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	age_na	bp_na	bgr_na	bu_na	sc_na	sod_na	pot_na	hemo_na	pcv_na	wc_na	rc_na
0	0.522727	0.230769	0.8	0.333333	0.166667	0.0	1.0	0.5	0.5	0.211538	0.088575	0.010582	0.842271	0.042697	0.836735	0.777778	0.231405	0.525424	1.0	1.0	0.5	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
1	0.056818	0.000000	0.8	0.833333	0.166667	0.0	1.0	0.5	0.5	0.211538	0.042362	0.005291	0.842271	0.042697	0.557823	0.644444	0.157025	0.457627	0.5	0.5	0.5	1.0	0.5	0.5	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0
2	0.681818	0.230769	0.4	0.500000	0.666667	1.0	1.0	0.5	0.5	0.856838	0.132221	0.018519	0.842271	0.042697	0.442177	0.488889	0.219008	0.457627	0.5	1.0	0.5	0.5	0.5	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0
3	0.522727	0.153846	0.2	0.833333	0.166667	1.0	0.5	1.0	0.5	0.202991	0.139923	0.044974	0.671924	0.000000	0.551020	0.511111	0.185950	0.305085	1.0	0.5	0.5	0.5	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.556818	0.230769	0.4	0.500000	0.166667	1.0	1.0	0.5	0.5	0.179487	0.062901	0.013228	0.842271	0.042697	0.578231	0.577778	0.210744	0.423729	0.5	0.5	0.5	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0

3.2 Creating Training set and Test set

In [34]:
```python
x_train, x_test, y_train, y_test = train_test_split(x_features_scaled, response, test_size=0.25, stratify=response, random_state=42)
```

In [35]:
```python
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[35]:
```
((300, 35), (300,), (100, 35), (100,))
```

In [36]:
```python
x_train.head()
```

Out[36]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	age_na	bp_na	bgr_na	bu_na	sc_na	sod_na	pot_na	hemo_na	pcv_na	wc_na	rc_na
67	0.488636	0.230769	0.8	0.666667	0.166667	1.0	0.5	0.5	0.5	0.861111	0.103979	0.011905	0.842271	0.042697	0.649660	0.688889	0.239669	0.457627	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
180	0.806818	0.307692	0.4	0.333333	0.833333	0.5	0.5	1.0	0.5	0.452991	0.139923	0.019841	0.842271	0.042697	0.489796	0.422222	0.239669	0.457627	0.5	1.0	0.5	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0
234	0.397727	0.384615	0.4	0.166667	0.166667	0.5	1.0	0.5	0.5	0.211538	0.044929	0.011905	0.842271	0.042697	0.809524	0.777778	0.078512	0.525424	1.0	0.5	0.5	1.0	0.5	0.5	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
243	0.681818	0.307692	0.8	0.500000	0.333333	0.0	1.0	0.5	0.5	0.314103	0.119384	0.026455	0.842271	0.008989	0.700680	0.844444	0.363636	0.677966	1.0	0.5	0.5	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
147	0.659091	0.076923	0.4	0.666667	0.333333	1.0	0.5	1.0	0.5	0.568376	0.088575	0.017196	0.791798	0.011236	0.326531	0.355556	0.537190	0.152542	1.0	0.5	0.5	0.5	0.5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [37]:
```python
x_test.head()
```

Out[37]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	age_na	bp_na	bgr_na	bu_na	sc_na	sod_na	pot_na	hemo_na	pcv_na	wc_na	rc_na
70	0.670455	0.230769	0.6	0.166667	0.833333	0.0	1.0	0.5	0.5	0.722222	0.044929	0.003968	0.835962	0.042697	0.823129	0.777778	0.252066	0.525424	1.0	1.0	0.5	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
38	0.761364	0.230769	0.8	0.666667	0.166667	0.5	1.0	0.5	0.5	0.211538	0.260591	0.048942	0.804416	0.076404	0.639456	0.688889	0.239669	0.457627	1.0	0.5	0.5	1.0	0.5	0.5	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0
27	0.761364	0.153846	0.4	0.666667	0.833333	1.0	0.5	0.5	0.5	0.517094	0.219512	0.030423	0.791798	0.033708	0.639456	0.622222	0.305785	0.338983	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
395	0.602273	0.230769	0.8	0.166667	0.166667	1.0	1.0	0.5	0.5	0.252137	0.121951	0.001323	0.917981	0.053933	0.857143	0.844444	0.185950	0.474576	0.5	0.5	0.5	1.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
85	0.772727	0.153846	0.6	0.500000	0.000000	0.0	0.0	0.5	0.5	0.211538	0.114249	0.014550	0.842271	0.042697	0.462585	0.688889	0.239669	0.457627	0.5	1.0	0.5	0.5	1.0	0.5	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0

In [38]:
```python
y_train.head()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-38-21e7fc29364a> in <module>
----> 1 y_train.head()

AttributeError: 'numpy.ndarray' object has no attribute 'head'
```

In [ ]:
```python
y_test
```

In [ ]:
```python
# Remove the augmented features during impuatation
x_train.drop(x_train.iloc[:,24:], axis=1, inplace=True)
x_test.drop(x_test.iloc[:,24:], axis=1, inplace=True)
```

In [ ]:
```python
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

In [ ]:
```python
x_train.head()
```

In [ ]:
```python
x_test.head()
```

4. Model Building

In [ ]:
```python
score = []
model = []
```

In [ ]:
```python
# utility function to print cross k-fold cross validation scores, their mean and variance(standard deviation)
def print_scores(model, x_train, y_train, cv, scoring):
    print('Cross validation scores:', cross_val_score(model, x_train.values, y_train, cv=cv, scoring=scoring, n_jobs=-1) )
    print( 'Mean_scores:', np.mean( cross_val_score(model, x_train.values, y_train, cv=cv, scoring=scoring, n_jobs=-1) ) )
    print( 'Variance:', np.std( cross_val_score(model, x_train.values, y_train, cv=cv, scoring=scoring, n_jobs=-1) ) )
```

4.1 Random Forest Classifier

In [ ]:
```python
random_forest = RandomForestClassifier(random_state=42, n_jobs=-1)
print_scores(random_forest, x_train, y_train, 5, 'accuracy')
```

In [ ]:
```python
random_forest.fit(x_train.values, y_train)
```

In [ ]:
```python
random_forest_preds = random_forest.predict(x_test.values)
```

In [ ]:
```python
random_forest_score = met.accuracy_score(y_test, random_forest_preds) * 100
score.append(random_forest_score)
model.append('Random Forest Classifier')
```

In [ ]:
```python
print('Random Forest Classifier Accuracy = ', random_forest_score )
```

In [ ]:
```python
cf_matrix = confusion_matrix(y_test, random_forest_preds)

print('Condusion Matrix:',cf_matrix)
```

In [ ]:
```python
#Utility function for plotting confusion matrix

def plot_cm(y_true, y_pred):
    cm = met.confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel("y_pred")
    plt.ylabel("y_true")
    plt.show()
```

In [ ]:
```python
plot_cm(y_test, random_forest_preds)
```

5. Saving a Model for Deployment

In [ ]:
```python

```

In [ ]:
```python
joblib.dump(scaler, './model/minmax_ckd.pkl')
```

In [ ]:
```python
joblib.dump(random_forest, './model/random_forest_ckd.pkl')
```

In [ ]:
```python
# Loading the saved model for prediction
```

In [ ]:
```python
random_forest_load = joblib.load('./model/random_forest_ckd.pkl')
```

In [ ]:
```python
prediction = random_forest_load.predict(x_test)
```

In [ ]:
```python
prediction== random_forest_preds
```

6. Feature Important

In [ ]:
```python
x_features_scaled.head()
```

In [ ]:
```python
features=x_features_scaled.iloc[:, :24]
features
```

In [ ]:
```python
response
```

In [ ]:
```python
cat_col
```

In [ ]:
```python
num_col
```

In [ ]:
```python

```

In [ ]:
```python
df=df.astype('string').fillna('0')
```

In [ ]:
```python
le=LabelEncoder()
```

In [ ]:
```python
for col in cat_col:
  df[col]=le.fit_transform(df[col])
```

In [ ]:
```python
ind_col=[col for col in df.columns if col!='class']
dep_col='class'
```

In [ ]:
```python
features=df[ind_col]
response=df[dep_col]
```

In [ ]:
```python
imp_features=SelectKBest(score_func=chi2,k=20)
imp_features=imp_features.fit(features,response)
```

In [ ]:
```python
imp_features
```

In [ ]:
```python
imp_features.scores_
```

In [ ]:
```python
datascore=pd.DataFrame(imp_features.scores_,columns=['Score'])
dfcols=pd.DataFrame(features.columns)
dfcols
```

In [ ]:
```python
features_rank=pd.concat([dfcols,datascore],axis=1)
features_rank.columns=['features','score']
features_rank
```

In [ ]:
```python
columns=pd.read_csv("./input/data_description.txt",sep='-')
columns=columns.reset_index(drop=True)
columns
```

In [ ]:
```python
features_rank.features=columns.cols
features_rank
```

In [ ]:
```python
imp_features_sorted=features_rank.nlargest(24,'score')
imp_features_sorted.to_csv(r'./Sample_results/feature_important_with_score.csv', index=False)
```

In [ ]:
```python
important_feature_names=imp_features_sorted['features']
important_feature_names.to_csv(r'./model/feature_important.csv', index=False)
```

In [ ]:
```python

```

In [ ]:
```python

```