Milestone2:

```
import pandas as pd
import numpy as np
from collections import Counter as c
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
```

```
data=pd.read_csv('/content/kidney_disease[1].csv')
data.head()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	w
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	780
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	600
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	750
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	670
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	730

5 rows × 26 columns

```
data.columns
```

```
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.columns=['age','blood_pressure','specific_gravity','albumin',
              'sugar','red_blood_cells','pus_cell','pus_cell_clumps','bacteria',
               'blood glucose random','blood_urea','serum_creatine','sodium','potassium',
               'hemoglobin','packed_cell_volume','white_blood_cell_count','red_blood_cell_count',
               'hypertension','diabetesmellitus','coronary_artery_disease','appetite',
               'pedal_edema','anemia','class']
data.columns
```

```
Index(['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
       'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'blood glucose random', 'blood_urea', 'serum_creatine', 'sodium',
       'potassium', 'hemoglobin', 'packed_cell_volume',
       'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
       'diabetesmellitus', 'coronary_artery_disease', 'appetite',
       'pedal_edema', 'anemia', 'class'],
      dtype='object')
```

```
data.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   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 random    356 non-null    float64
 10  blood_urea              381 non-null    float64
 11  serum_creatine          383 non-null    float64
```

```
 12  sodium                  313 non-null    float64
 13  potassium               312 non-null    float64
 14  hemoglobin              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  diabetesmellitus        398 non-null    object
 20  coronary_artery_disease 398 non-null    object
 21  appetite                399 non-null    object
 22  pedal_edema             399 non-null    object
 23  anemia                  399 non-null    object
 24  class                   400 non-null    object
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

```
data.isnull().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_creatine              17
sodium                      87
potassium                   88
hemoglobin                  52
packed_cell_volume          70
white_blood_cell_count     105
red_blood_cell_count       130
hypertension                 2
diabetesmellitus             2
coronary_artery_disease      2
appetite                     1
pedal_edema                  1
anemia                       1
class                        0
dtype: int64
```

```
data['blood glucose random'].fillna(data['blood glucose random'].mean(),inplace=True)
data['blood_pressure'].fillna(data['blood_pressure'].mean(),inplace=True)
data['blood_urea'].fillna(data['blood_urea'].mean(),inplace=True)
data['hemoglobin'].fillna(data['hemoglobin'].mean(),inplace=True)
data['packed_cell_volume'].fillna(data['packed_cell_volume'].mean(),inplace=True)
data['potassium'].fillna(data['potassium'].mean(),inplace=True)
data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mean(),inplace=True)
data['serum_creatine'].fillna(data['serum_creatine'].mean(),inplace=True)
data['sodium'].fillna(data['sodium'].mean(),inplace=True)
data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].mean(),inplace=True)
```

```
data['age'].fillna(data['age'].mode()[0],inplace=True)
data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)
data['pus_cell_clumps'].fillna(data['pus_cell_clumps'].mode()[0],inplace=True)
data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)
data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)
data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace=True)
data['red_blood_cells'].fillna(data['red_blood_cells'].mode()[0],inplace=True)
data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mode()[0],inplace=True)
data['bacteria'].fillna(data['bacteria'].mode()[0],inplace=True)
data['anemia'].fillna(data['anemia'].mode()[0],inplace=True)
data['sugar'].fillna(data['sugar'].mode()[0],inplace=True)
data['diabetesmellitus'].fillna(data['diabetesmellitus'].mode()[0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)
data['specific_gravity'].fillna(data['specific_gravity'].mode()[0],inplace=True)
```

```
contcols=set(data.dtypes[data.dtypes!='0'].index.values)
#contcols=pd.DataFrame(data,columns=contcols)
print(contcols)
```

```
{'sg', 'rbc', 'id', 'pot', 'bgr', 'pcc', 'su', 'sc', 'pe', 'wc', 'rc', 'ba', 'sod', 'classification', 'pcv', 'pc', 'age', 'dm', 'ht
```

```
for i in catcols:
  print("Columnns:",i)
```

```
  print(c(data[i]))
  print('*'*120+'\n')
```

```
    Columnns: sugar
    Counter({0.0: 339, 2.0: 18, 3.0: 14, 4.0: 13, 1.0: 13, 5.0: 3})
    ************************************************************************************************************************

    Columnns: specific_gravity
    Counter({1.02: 153, 1.01: 84, 1.025: 81, 1.015: 75, 1.005: 7})
    ************************************************************************************************************************

    Columnns: albumin
    Counter({0.0: 245, 1.0: 44, 2.0: 43, 3.0: 43, 4.0: 24, 5.0: 1})
    ************************************************************************************************************************
```

```
#'specific_gravity','albumin','sugar'(as these columns are numerical it is removed)
catcols=['anemia','pedal_edma','appetite','bacteria','class','coronary_artery_disease','diabetesmellit',
         'hypertension','pus_cell','pus_cell_clumps','red_blood_cells']
```

```
from sklearn.preprocessing import LabelEncoder
for i in catcols:
  print("LABEL ENCODING OF:",i)
  LEi=LabelEncoder()
  print(c(data[i]))
  data[i] = LEi.fit_transform(data[i])
  print(c(data[i]))
  print("*"*100)
```

```
contcols=set(data.dtypes[data.dtypes!='0'].index.values)
#contcols=pd.DataFrame(data,columns=contcols)
print(contcols)
```

```
    {'sg', 'rbc', 'id', 'pot', 'bgr', 'pcc', 'su', 'sc', 'pe', 'wc', 'rc', 'ba', 'sod', 'classification', 'pcv', 'pc', 'age', 'dm', 'ht
```

```
for i in contcols:
  print("Continous Columns:",i)
  print(c(data[i]))
  print('*'*120+'\n')
```

```
    ************************************************************************************************************************
```

```
**********************************************************************************************************************

Continous Columns: al
Counter({0.0: 199, 1.0: 44, 2.0: 43, 3.0: 43, 4.0: 24, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, n
**********************************************************************************************************************

Continous Columns: bp
Counter({80.0: 116, 70.0: 112, 60.0: 71, 90.0: 53, 100.0: 25, 50.0: 5, 110.0: 3, nan: 1, nan: 1, 140.0: 1, 180.0: 1, nan: 1, nan
**********************************************************************************************************************

Continous Columns: appet
Counter({'good': 317, 'poor': 82, nan: 1})
```

```
contcols.remove('specific_gravity')
contcols.remove('albumin')
contcols.remove('sugar')
print(contcols)
```

```
{'blood_urea', 'pus_cell_clumps', 'red_blood_cells', 'coronary_artery_disease', 'appetite', 'potassium', 'pedal_edema', 'anemia',
```

```
contcols.add('red_blood_cell_count')
contcols.add('packed_cell_volume')
contcols.add('white_blood_cell_count')
print(contcols)
```

```
{'sg', 'rbc', 'id', 'pot', 'bgr', 'pcc', 'su', 'sc', 'pe', 'wc', 'rc', 'ba', 'white_blood_cell_count', 'red_blood_cell_count', 'pac
```

```
catcols.add('specific_gravity')
catcols.add('albumin')
catcols.add('sugar')
print(catcols)
```

```
{'sugar', 'specific_gravity', 'albumin'}
```

```
data['coronary_artery_disease'] = data.coronary_artery_disease.replace('\tno','no')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-183-b86ea066d7e2> in <cell line: 1>()
----> 1 data['coronary_artery_disease'] = dataset.coronary_artery_disease.replace('\tno','no')

NameError: name 'dataset' is not defined
```

SEARCH STACK OVERFLOW

```
data['diabetesmellitus'] = data.diabetesmellitus.replace(to_replace={'\tno':'no','\types':'yes','yes':})
c(data[diabetesmellitus])
```

```
  File "<ipython-input-180-d85ff3d59297>", line 1
    data['diabetesmellitus'] = data.diabetesmellitus.replace(to_replace={'\tno':'no','\types':'yes','yes':})
                                                                                                           ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

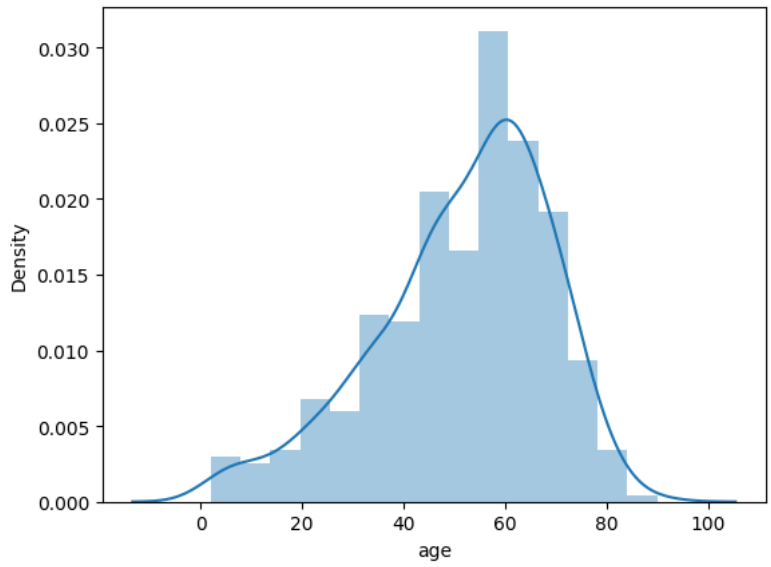
milestone3

```
data.describe()
```

	age	blood_pressure	specific_gravity	albumin	sugar	blood glucose random	blood_urea	serum_creatine	sodium	pota
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.6

```
sns.distplot(data.age)
```

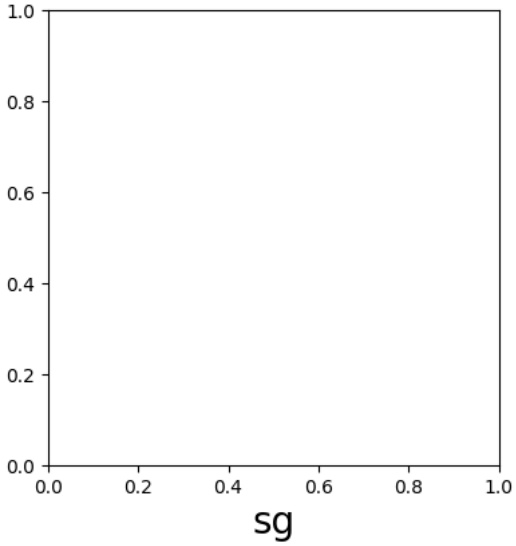
```
<ipython-input-45-868c85374ad7>:1: UserWarning:

`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.age)
<Axes: xlabel='age', ylabel='Density'>
```

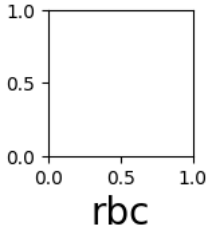


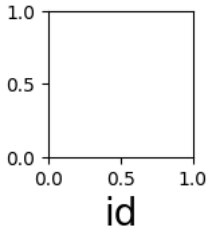

```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(5,5))
plt.scatter(data['age'],data['blood_pressure'],color='blue')
plt.xlabel('age')
plt.ylabel('blood pressure')
plt.title("age vs blood scatter plot")
```

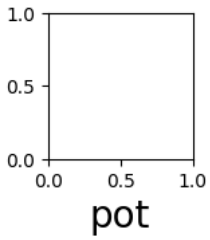
```
Text(0.5, 1.0, 'age vs blood scatter plot')
```

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

for column in contcols:
  if plotnumber<=11:
    ax=plt.subplot(3,4,plotnumber)

    plt.xlabel(column,fontsize=20)

    plotnumber+=1
    plt.show()
```










```
#HEAT MAP #correlation of parameters
f,ax=plt.subplots(figsize=(18,10))
sns.heatmap(data.corr(),annot=True,fmt=".2f",ax=ax,linewidths=0.5,linecolor="orange")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```

```
<ipython-input-147-9d5ae916c1e3>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future v
  sns.heatmap(data.corr(),annot=True,fmt=".2f",ax=ax,linewidths=0.5,linecolor="orange")
```


```
sns.countplot(data['class'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3801             try:
-> 3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:

 4 frames 

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'class'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:
-> 3804                 raise KeyError(key) from err
   3805             except TypeError:
   3806                 # If we have a listlike key, _check_indexing_error will raise

KeyError: 'class'
```

SEARCH STACK OVERFLOW

```
# perfroming feature scaling operation using standard scaller on X part of the dataset because
# there different type of values in the columns
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_bal=sc.fit_transform
```

```
selcols=['red_blood_cells','pus_cell','blood glucose random','blood_urea',
         'pedal_edema','anemia','diabetesmellitus','coronary_artery_disease']
x=pd.DataFrame(data,columns=selcols)
y=pd.DataFrame(data,columns=['class'])
print(x.shape)
print(y.shape)
```

```
(400, 8)
(400, 1)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(320, 8)
(320, 1)
(80, 8)
(80, 1)
```

Milestone4

```
#importing the keras libraries and packages
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


#creating ANN skleton view
classification = Sequential()
classification.add(Dense(30,activation='relu'))
classification.add(Dense(128,activation='relu'))
classification.add(Dense(64,activation='relu'))
classification.add(Dense(1,activation='sigmoid'))


#Compiling the ANN model
classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])


classification.fit(x_train,y_train,batch_size=10,validation_split=0.3,epochs=100)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-178-85d4caec70a2> in <cell line: 1>()
----> 1 classification.fit(x_train,y_train,batch_size=10,validation_split=0.3,epochs=100)

                                 1 frames
/usr/local/lib/python3.9/dist-packages/tensorflow/python/framework/constant_op.py in convert_to_eager_tensor(value, ctx, dtype)
    101       dtype = dtypes.as_dtype(dtype).as_datatype_enum
    102   ctx.ensure_initialized()
--> 103   return ops.EagerTensor(value, ctx.device_name, dtype)
    104
    105

ValueError: Failed to convert a NumPy array to a Tensor (Unsupported object type float).
```

SEARCH STACK OVERFLOW

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')


rfc.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-165-b87bb2ba9825> in <cell line: 1>()
----> 1 rfc.fit(x_train,y_train)

                                 5 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in __array__(self, dtype)
   2068
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071
   2072     def __array_wrap__(

ValueError: could not convert string to float: 'normal'
```

SEARCH STACK OVERFLOW

```
y_predict = rfc.predict(x_test)
```

```
y_predict_train = rfc.predict(x_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-159-6d0b7f55f6ea> in <cell line: 1>()
----> 1 y_predict_train = rfc.predict(x_train)

 6 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in __array__(self, dtype)
   2068
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071
   2072     def __array_wrap__(

ValueError: could not convert string to float: 'normal'
```

SEARCH STACK OVERFLOW

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(max_depth=4,splitter='best,criterion='entropy')
```

```
  File "<ipython-input-160-aa38db87d527>", line 2
    dtc = DecisionTreeClassifier(max_depth=4,splitter='best,criterion='entropy')
                                                                               ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
dtc.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-161-8bc0353a9ffd> in <cell line: 1>()
----> 1 dtc.fit(x_train,y_train)

NameError: name 'dtc' is not defined
```

SEARCH STACK OVERFLOW

```
y_predict= dtc.predict(x_test)
y_predict
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-162-b3c3150919dc> in <cell line: 1>()
----> 1 y_predict= dtc.predict(x_test)
      2 y_predict

NameError: name 'dtc' is not defined
```

SEARCH STACK OVERFLOW

```
y_predict_train = dtc.predict(x_train)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-163-4e71ba4a880d> in <cell line: 1>()
----> 1 y_predict_train = dtc.predict(x_train)

NameError: name 'dtc' is not defined
```

SEARCH STACK OVERFLOW

```
from sklearn.linear_model import LogisticRegression



lgr = LogisticRegression()


lgr.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-169-0be1d0fe3673> in <cell line: 1>()
----> 1 lgr.fit(x_train,y_train)

 5 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in __array__(self, dtype)
   2068
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071
   2072     def __array_wrap__(

ValueError: could not convert string to float: 'normal'
```

```
from sklearn.metrics import accuracy_score,classification_report
```

```
y_predict = lgr.predict(x_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-171-361d7e5c1e35> in <cell line: 1>()
----> 1 y_predict = lgr.predict(x_test)

 5 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in __array__(self, dtype)
   2068
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071
   2072     def __array_wrap__(

ValueError: could not convert string to float: 'normal'

SEARCH STACK OVERFLOW
```

```
y_pred = lgr.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
y_pred
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegressic
  warnings.warn(
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-172-0bf88aa12548> in <cell line: 1>()
----> 1 y_pred = lgr.predict([[1,1,121.000000,36.0,0,0,1,0]])
      2 print(y_pred)
      3 y_pred

 1 frames
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_base.py in decision_function(self, X)
    399
    400         X = self._validate_data(X, accept_sparse="csr", reset=False)
--> 401         scores = safe_sparse_dot(X, self.coef_.T, dense_output=True) + self.intercept_
    402         return xp.reshape(scores, -1) if scores.shape[1] == 1 else scores
    403

AttributeError: 'LogisticRegression' object has no attribute 'coef_'

SEARCH STACK OVERFLOW
```

```
y_pred = dtc.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_predy)
y_predict_train
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-173-d8b5d3f4ae4d> in <cell line: 1>()
----> 1 y_pred = dtc.predict([[1,1,121.000000,36.0,0,0,1,0]])
      2 print(y_predy)
      3 y_predict_train

NameError: name 'dtc' is not defined

SEARCH STACK OVERFLOW
```

```
y_pred = rfc.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
```

```
y_pred
```

```
classification.save("ckd.h5")
```

```
y_pred = classification.predict(x_test)
```

```
y_pred
```

```
y_pred = (y_pred > 0.5)
y_pred
```

```
def predict_exit(sample_value):
  # Convert list to numpy array
  sample_value = np.array(sample_value)
  #Reshape because sample_value contains only 1 record
  sample_value = sample_value.reshape(1,-1)
  #Feature Scaling
  sample_value = sc,transform(sample_value)
  return classifier.predict(sample_value)
```

```
test=classification.predict([[1,1,121,000000,36,0,0,0,1,0]])
if test==1:
  print('Prediction: Highh chance of CKD!')
else:
  print('Prediction:Low chance of CKD.')
```

milestone5

```
from sklearn import model_selection
```

```
dfs =[]
models =[
    ('LogReg',LogisticRegression()),
    ('RF',RandomForestClassifier()),
    ('DecisionTree',DecisionTreeClassifier)),
]
results =[]
 names =[]
 scoring =['accuracy','precision_weighted','recall_weighted','f1_weighted','roc_auc']
 target_names =['No CKD','CKD']
 for name, model in models:
   kfold=model_selection.kFkkold(n_splits=5,shuffle=True,random_state=90210)
   cv_results = model_selection.cross_validate(model,x_train,y_train,cv=kfold,scoring=scoring)
   clf = model.fit(x_train,y_train)
   y_pred =clf.predict(x_test)
   print(name)
   print(classification_report(y_test,y_pred,target_names=target_names))
   results.append(cv_results)
   names.append(name)
   this_df = pd.DataFrame(cv_results)
   this_df['model'] = name
   dfs.append(this_df)
final = pd.concat(dfs,ignore_index=True)
return final
```

```
       File "<ipython-input-132-e9b81462d286>", line 5
         ('DecisionTree',DecisionTreeClassifier)),
                                                 ^
     SyntaxError: closing parenthesis ')' does not match opening parenthesis '[' on line 2
```

SEARCH STACK OVERFLOW

```
from sklearn.metrics import confision_matrix
cm = confusion_matrix(y_test,y_predict)
cm
```

```
# Plotting confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm.cmap='Blues',annot=True,xticklabels=['no ckd','ckd'],yticklabels=['no,ckd','ckd'])
plt.xlabel('predicted values')
plt.ylabel('Actual values')
plt.title('Confusion Matrix for Logistic Regression model')
plt.show()
```

```
from sklearn.metrics import confision_matrix
cm = confusion_matrix(y_test,y_predict)
cm
```

```
# Plotting confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm.cmap='Blues',annot=True,xticklabels=['no ckd','ckd'],yticklabels=['no,ckd','ckd'])
plt.xlabel('predicted values')
plt.ylabel('Actual values')
plt.title('Confusion Matrix for Logistic Regression model')
plt.show()
```

```
from sklearn.metrics import confision_matrix
cm = confusion_matrix(y_test,y_predict)
cm
```

```
# Plotting confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm.cmap='Blues',annot=True,xticklabels=['no ckd','ckd'],yticklabels=['no,ckd','ckd'])
plt.xlabel('predicted values')
plt.ylabel('Actual values')
plt.title('Confusion Matrix for Logistic Regression model')
plt.show()
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving kidney_disease[1].csv to kidney_disease[1] (1).csv

```
from sklearn.metrics import confision_matrix
cm = confusion_matrix(y_test,y_predict)
cm
```

```
# Plotting confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm.cmap='Blues',annot=True,xticklabels=['no ckd','ckd'],yticklabels=['no,ckd','ckd'])
plt.xlabel('predicted values')
plt.ylabel('Actual values')
plt.title('Confusion Matrix for Logistic Regression model')
plt.show()
```

```
bootstaps -[]
for model in list(set(final.model.values)):
  model_df = final.loc[final.model == model]
  bootstrap = madel_df.sample(n=30, replace=True)
  bootstraps.append(bootstrap)
bootstrap_df = pd.concat(bootstraps,ignore_ignore_index=True)
results_long = pd.melt(bootstrap_df,id_vars=['model'],var_name='metrics',value_name='values')
time_metrics+['fit_time','score_time']
## PERFORMANCE METRICS
results_long_nofit = results_long.loc[~results_long['metrics'].isin(time_matrics)]
results_long_nofit = results_long_nofit.sort_values(by='values')
```

```
## TIME METRICS
results_long_nofit = results_long.loc[~results_long['metrics'].isin(time_matrics)]
results_long_nofit = results_long_nofit.sort_values(by='values')
```

```
import matplotlib.pyplot as plt
import seaborn as sns'
plt.figure(figsize=(20,12))
sns.set(font_scale=2.5)
g = sns.boxplot(x="model",y="values",hue="metrics",data=results_long_nofit,palette="Set3")
plt.legend(bbox_to_anchor=(1.05,1), loc=2, borderxespad=0)
plt.title("Comparison of Model by Classification Metric")
plt.savefig('./benchmark_models_performance.png',dpi=300)
```

Milestone6

```
pickle.dump(lgr, open('CKD.pk1','wb'))
```

```
data.tail()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classifica
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47	6700	4.9	no	no	no	good	no	no	n
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54	7800	6.2	no	no	no	good	no	no	n
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49	6600	5.4	no	no	no	good	no	no	n
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51	7200	5.9	no	no	no	good	no	no	n
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53	6800	6.1	no	no	no	good	no	no	n

5 rows × 26 columns

```
data.head(10)
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	clas
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	
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	
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	
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	
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	
5	5	60.0	90.0	1.015	3.0	0.0	NaN	NaN	notpresent	notpresent	...	39	7800	4.4	yes	yes	no	good	yes	no	
6	6	68.0	70.0	1.010	0.0	0.0	NaN	normal	notpresent	notpresent	...	36	NaN	NaN	no	no	no	good	no	no	
7	7	24.0	NaN	1.015	2.0	4.0	normal	abnormal	notpresent	notpresent	...	44	6900	5	no	yes	no	good	yes	no	
8	8	52.0	100.0	1.015	3.0	0.0	normal	abnormal	present	notpresent	...	33	9600	4.0	yes	yes	no	good	no	yes	
9	9	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	...	29	12100	3.7	yes	yes	no	poor	no	yes	

10 rows × 26 columns

```
data.drop(["id"],axis=1,inplace=True)
```

```
data['class'].unique()
```

```
array(['ckd', 'ckd\t', 'notckd'], dtype=object)
```

```
data['class']=data['class'].replace("ckd\t","ckd")
data['class'].unique()
```

```
array(['ckd', 'notckd'], dtype=object)
```

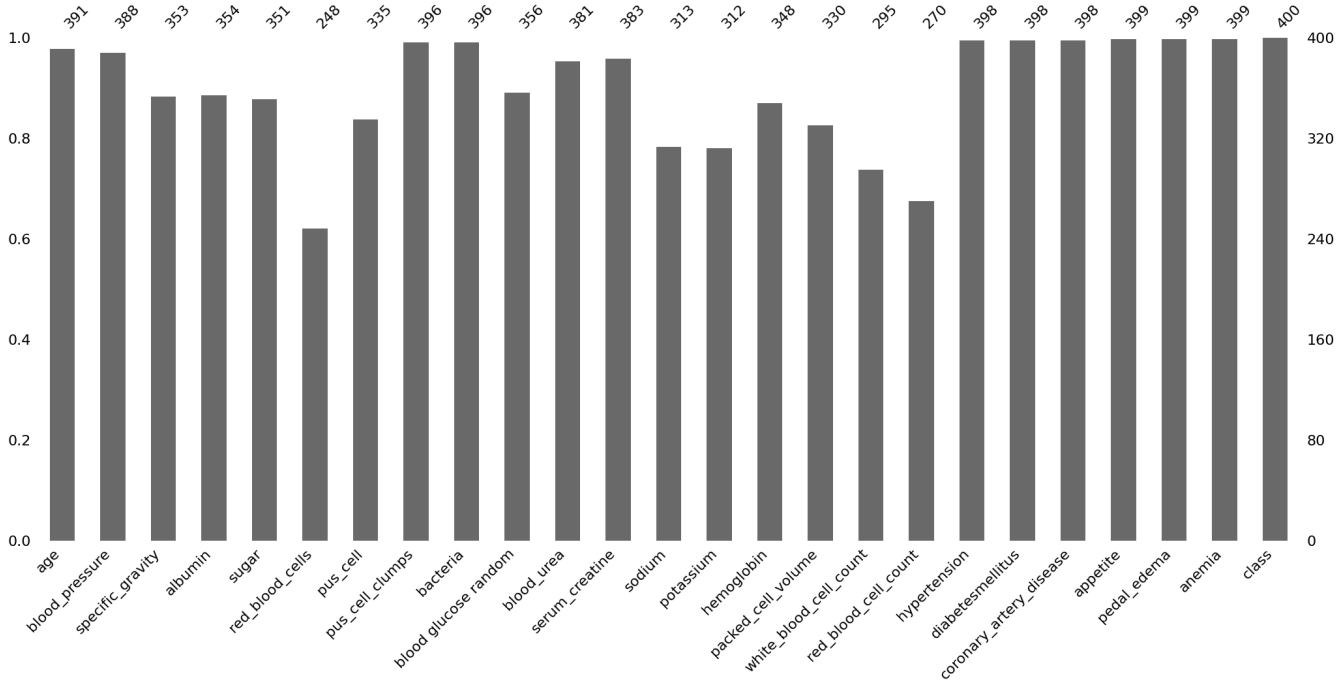
```
np.unique(data.dtypes,return_counts=True)
```

```
(array([dtype('float64'), dtype('O')], dtype=object), array([11, 14]))
```

```
data.isnull().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_creatine              17
sodium                      87
potassium                   88
hemoglobin                  52
packed_cell_volume          70
white_blood_cell_count     105
red_blood_cell_count       130
hypertension                 2
diabetesmellitus             2
coronary_artery_disease      2
appetite                     1
pedal_edema                  1
anemia                       1
class                        0
dtype: int64
```

```
#sns.heatmap(data.isnull(), cbar=False)
msno.bar(data)
plt.show()
```




```
data.packed_cell_volume = pd.to_numeric(data.packed_cell_volume,errors='coerce')
data.white_blood_cell_count = pd.to_numeric(data.white_blood_cell_count,errors='coerce')
data.red_blood_cell_count = pd.to_numeric(data.red_blood_cell_count,errors='coerce')
```

```
data.isnull().sum()
```

```
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_creatine             0
sodium                     0
potassium                  0
hemoglobin                 0
packed_cell_volume         0
```

```
white_blood_cell_count     0
red_blood_cell_count       0
hypertension               0
diabetesmellitus           0
coronary_artery_disease    0
appetite                   0
pedal_edema                0
anemia                     0
class                      0
dtype: int64
```

```
data['class'].unique()
```

```
array(['ckd', 'notckd'], dtype=object)
```

```
selcols=['red_blood_cells','pus_cell','blood glucose random','blood_urea',
         'pedal_edema','anemia','diabetesmellitus','coronary_artery_disease']
x=pd.DataFrame(data,columns=selcols)
y=pd.DataFrame(data,columns=['class'])
print(x.shape)
print(y.shape)
```

```
(400, 8)
(400, 1)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(320, 8)
(320, 1)
(80, 8)
(80, 1)
```

```
from sklearn.preprocessing import LabelEncoder
for i in catcols:
  print("LABEL ENCODING OF:",i)
  LEi=LabelEncoder()
  print(c(data[i]))
  data[i]=LEi.fit_transform(data[i])
  print(c(data[i]))
  print("*"*100)
```

```
data.describe()
```

	age	blood_pressure	specific_gravity	albumin	sugar	blood glucose random	blood_urea	serum_creatine	sodium	potas
count	400.000000	400.000000	400.000000	400.00000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00
mean	51.675000	76.469072	1.017712	0.90000	0.395000	148.036517	57.425722	3.072454	137.528754	4.62
std	17.022008	13.476298	0.005434	1.31313	1.040038	74.782634	49.285887	5.617490	9.204273	2.81
min	2.000000	50.000000	1.005000	0.00000	0.000000	22.000000	1.500000	0.400000	4.500000	2.50
25%	42.000000	70.000000	1.015000	0.00000	0.000000	101.000000	27.000000	0.900000	135.000000	4.00
50%	55.000000	78.234536	1.020000	0.00000	0.000000	126.000000	44.000000	1.400000	137.528754	4.62
75%	64.000000	80.000000	1.020000	2.00000	0.000000	150.000000	61.750000	3.072454	141.000000	4.80
max	90.000000	180.000000	1.025000	5.00000	5.000000	490.000000	391.000000	76.000000	163.000000	47.00