Early prediction for chronic kidney disease detection: a progressive approach to health management

INTRODUCTION

1.1OVERVIEW

- Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases.

- Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem, the predicted survival of the patient after the illness, the pattern of the disease and work for curing the disease.

- In todays world as we know most of the people are facing so many disease and as this can be cured if we treat people in early stages this project can use a pretrained model to predict the Chronic Kidney Disease which can help in treatments of peoples who are suffer from this disease.

- National kidney foundation classifies stages of CKD into five based on the abnormal kidney function and reduced Glomerular Filtration Rate (GFR), which measures a level of kidney function,. Te mildest stage (stage 1 and stage 2) is known with only a few symptoms and stage 5 is considered as end-stage or kidney failure. Te Renal Replacement Therapy (RRT) cost for total kidney failure is very expensive. Te

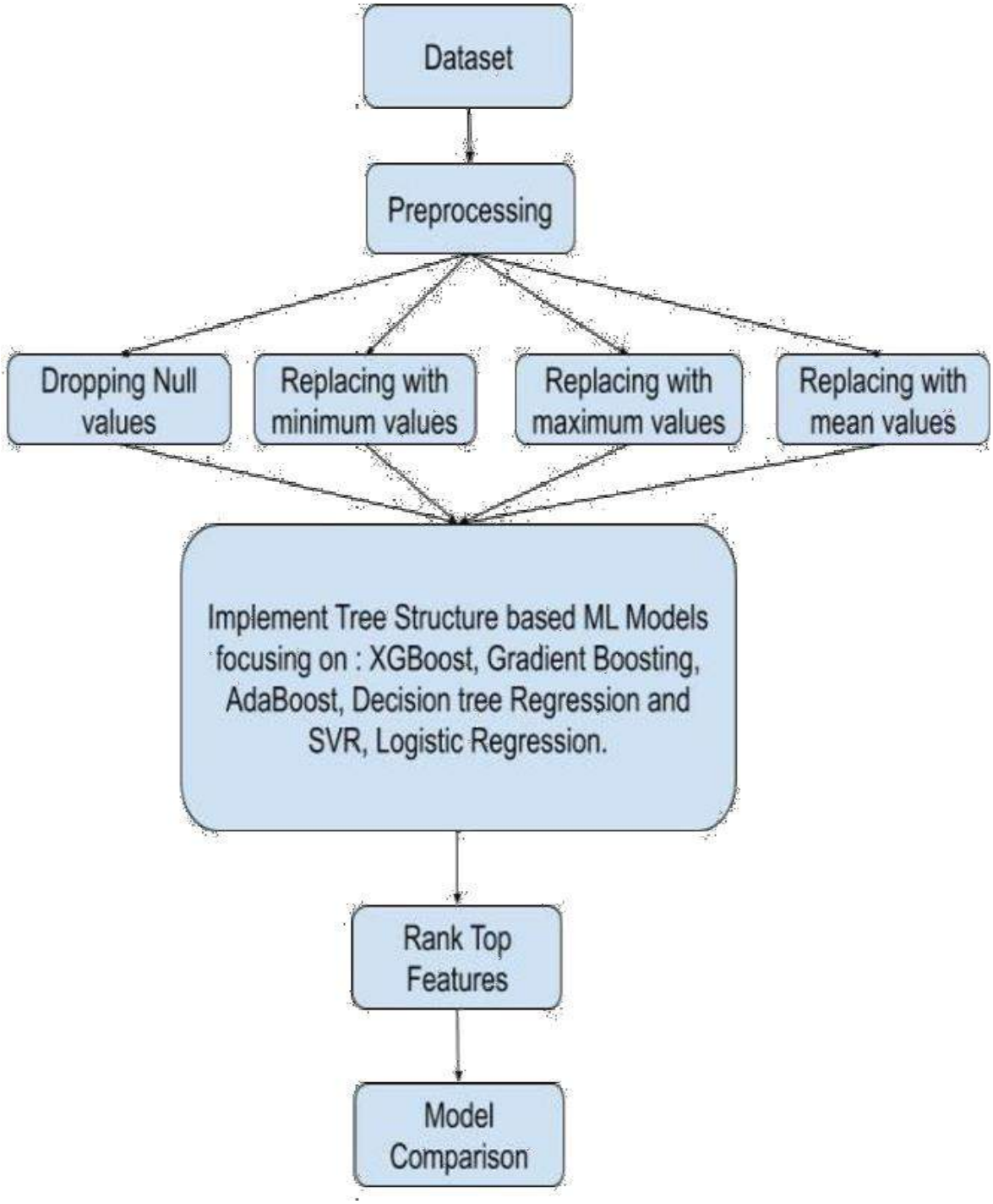
treatment is not also available in most developing countries like Ethiopia. As a result, the management of kidney failure and its complications is very diffcult in developing countries due to shortage of facilities, physicians, and the high cost to get the treatment

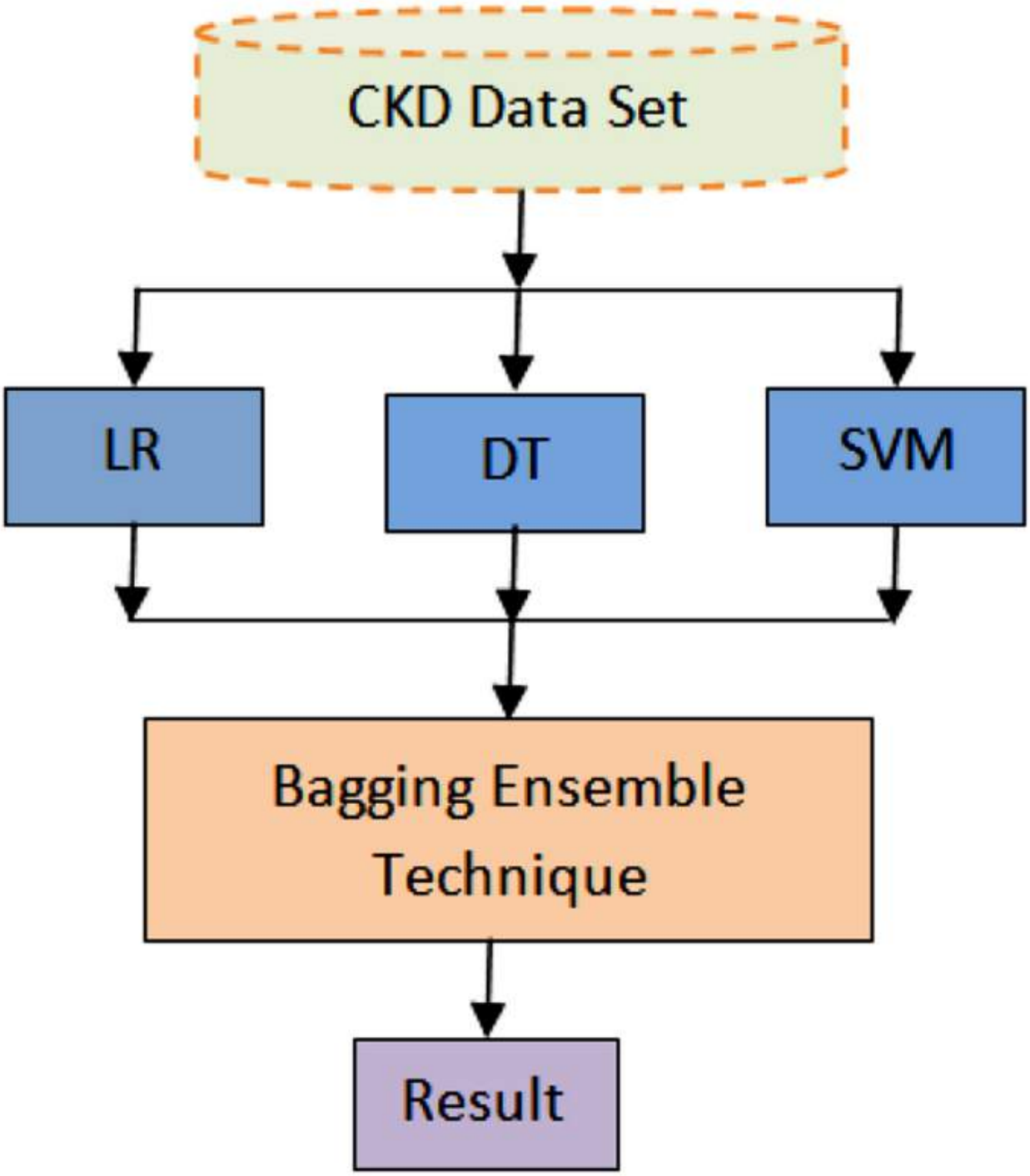
1.2PURPOSE

- Early prediction is very crucial for both experts and patients to prevent and slow down the progress of chronic kidney disease to kidney failure. In this study three machinelearning models RF, SV, DT, and two feature selection methods RFECV and UFS were used to build proposed models.
- The evaluation of models were done using tenfold crossvalidation. First, the four machine learning algorithms were applied to original datasets with all 19 features. Applying the models on the original dataset, we have got the highest accuracy with RF, SVM, and XGBoost.
- The accuracy was 99.8% for the binary class and 82.56% for five-class. DT produced lowest performance compared to RF. RF also produced the highest f1_score values. SVM and RF with RFECV produced the highest accuracy of 99.8%for binary class.
- XGBoost has 82.56% accuracy for fve-class datasets which is the highest. Hencewe believe that multi classification work was very important Debal and Sitote Journal of Big Data (2022) 9:109 Page 18 of 19 to know the stages of the disease and suggest needed treatments for the patients in order to save their lives.

2.problem definition & design thinking

2.1 empathy map



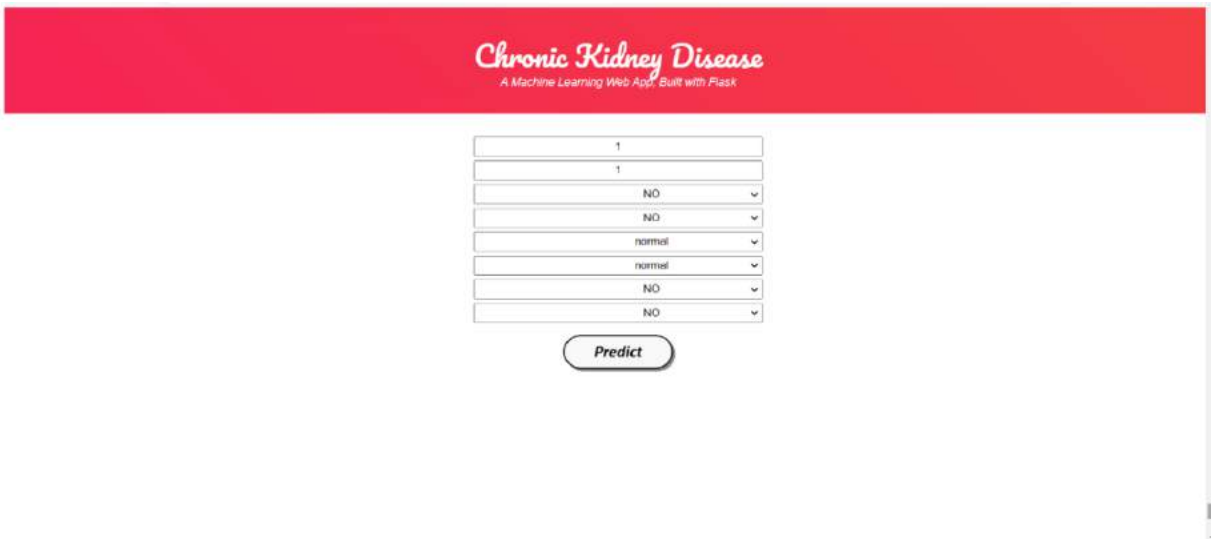


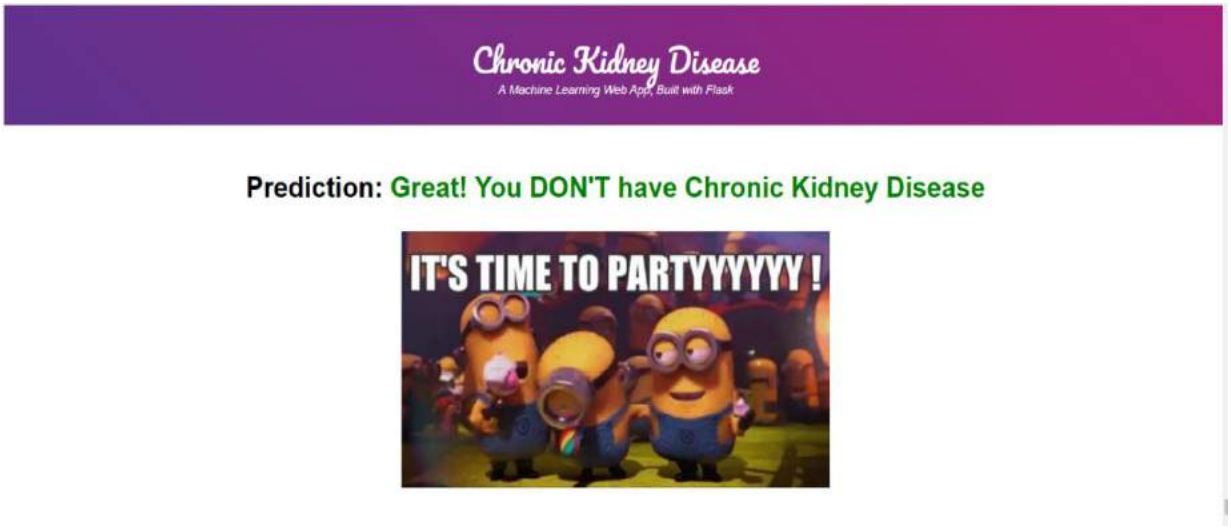
2.2 ideation & brainstorming map

3. Result

INPUT - NOW, THE USER WILL GIVE INPUTS TO GET THE PREDICTED RESULT AFTER CLICKIN GONTO THE SUBMIT BUTTN ON.

J


Chronic Kidney Disease
A Machine Learning Web App, Built with Flask
1
1
NO
NO
normal
normal
NO
NO
Predict


Chronic Kidney Disease
A Machine Learning Web App, Built with Flask
Prediction: Great! You DON'T have Chronic Kidney Disease
IT'S TIME TO PARTYYYYYY !

4. Trailhead profile public URL

Team leader –http://trailblazer.me/id/nit53

Team member 1 -http://trailblazer.me/id/naresh1907

Team member 2 -http://trailblazer.me/id/naresh1737

Team member 3 -http://trailblazer.me/id/keerb30

5. advantages & disadvantages

Advantages :

- National kidney foundation classifes stages of CKD into fve based on the abnormal kidney function and reduced Glomerular Filtration Rate (GFR), which measures a level of kidney function,.
- The mildest stage (stage 1 and stage 2) is known with only a few symptoms and stage 5 is considered as end-stage or kidney failure.
- Te Renal Replacement Terapy (RRT) cost for total kidney failure is very expensive. Te treatment is not also available in most developing countries like Ethiopia.
- As a result, the management of kidney failure and its complications is very difcult in developing countries due to shortage of facilities, physicians, and the high cost to get the treatment

Feature selection

Identify subset of relevant predictive features is important for quality result . Feature selection is the process of selecting most important predictive features to use them as input for models. It is important preprocessing step to deal with the problem of high dimensionality. Hence, the main aim of feature selection is to select the subset of features that are relevant and independent of each other for training the model . Similarly, feature selection is crucial to develop chronic kidney disease predictive model. This reduces the dimensionality and complexity of the data and makes the model be faster, more effective and accurate. Hence, feature selection algorithm have been used to select relevant features after the construction of the dataset

Disadvantages :

- Some of the common health problems caused by kidney disease include gout, anemia,secondary hyperparathyroidism(SHPT), bone disease,heart disease and fluid buildup.
- There are treatments to help manage health problems caused by kidney disease.

6.APPLICATIONS

Purpose of review :

The universal adoption of electronic health records, improvement in technology and the availability of continuous monitoring has generated large

quantities of health care data. Machine learning is increasingly adopted by nephrology researchers to analyze this data in order to improve the care of their patients.

Recent findings :

Researchers have demonstrated the ability of machine learning to read kidney biopsy samples, identify patient outcomes from unstructured data, identify subtypes in complex diseases, and discuss the potential benefits on drug discovery.

7. CONCLUSION

In this paper we have studied different machine learning algorithms. We have analysed 14 different attributes related to CKD patients and predicted accuracy for different machine learning algorithms like Decision tree and Support Vector Machine.

- From the results analysis, it is observed that the decision tree algorithms gives the accuracy of 91.75% and SVM gives accuracy of 96.75%. When considering the decision tree algorithm it builds the tree based on the entire dataset by using all the features of the dataset.
- The advantage of this system is that, the prediction process is less time consuming. It will help the doctors to start the treatments early for the CKD patients and also it will help to diagnose more patients within a less time period.
- Limitations of this study are the strength of the data is not higher because of the size of the data set and the missing attribute values. To build a machine learning model targeting chronic kidney disease with overall accuracy of 99.99%, will need millions of records with zero missing values.
- Early prediction is very crucial for both experts and patients to prevent and slow down the progress of chronic kidney disease to kidney failure. In this study three machinelearning models RF, SV, DT, and two feature selection methods RFECV and UFS were used to build proposed models.
- The evaluation of models were done using tenfold crossvalidation. First, the four machine learning algorithms were applied to original datasets with all 19 features. Applying the models on the original dataset, we have got the highest accuracy with RF, SVM, and XG Boost.
- The accuracy was 99.8% for the binary class and 82.56% for five-class. DT produced lowest performance compared to RF. RF also produced the highest f1_score values. SVM and RF with RFECV produced the highest accuracy of 99.8%for binary class. XG Boost has 82.56% accuracy for five-class datasets which is the highest. Hence we believe that multi classification work was very important know the stages of the disease and suggest needed treatments for the patients in order to save their lives

8.FUTURE SCOPE

This study used a supervised machine-learning algorithm, feature selection methods to select the best subset features to develop the models. It is better to see the difference in performance results using unsupervised or deep learning algorithms models. Te proposed model supports the experts to give the fast decision, it is better to make it a mobile-based system that enables the experts to follow the status of the patients and help the patients to use the system to know their status

Acknowledgements

Not applicable

Author contributions

The development of the basic research questions, identifying the problems and selecting appropriate machine learning algorithms, data collection, data analysis, interpretation, and critical review of the paper have been done by DA. The edition of the overall progress of the work was supported and guided by TM. Both authors read and approved the final manuscript.

Availability of data and materials

The data to conduct this research was collected from the patient history data attends their treatment or died during a period 2018 to 2019 from St. Paulo's Hospital. There are no restrictions on the availability of data and the authors are willing to provide the code as well.

Funding

Not applicable

SOURCE CODE :

Importing Libaries

```
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 library
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 #splits data in random train and test array
from sklearn.preprocessing import LabelEncoder #encoding the levels of categorical features
from sklearn.linear_model import LogisticRegression #Classification ML algorithm
import pickle #Python object hierarchy is converted into a byte stream,
```

```
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_creatinine'].fillna(data['serum_creatinine'].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)
```

Labeling Encoding of Categorical Column

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

```
from sklearn.preprocessing import LabelEncoder #imorting the LabelEncoding from sklearn
for i in catcols: #Looping through all the categorical columns
    print("LABEL ENCODING OF:",i)
    LEi = LabelEncoder() # creating an object of LabelEncoder
    print(c(data[i])) #getting the classes values before transformation
    data[i] = LEi.fit_transform(data[i])# trannsforming our text classes to numerical values
    print(c(data[i])) #getting the classes values after transformation
    print("*"*100)
```

In the above code we are looping through all the selected text class categorical columns and performing label encoding.

```
LABEL ENCODING OF: anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
****************************************************************************************************
LABEL ENCODING OF: pedal_edema
Counter({'no': 324, 'yes': 76})
Counter({0: 324, 1: 76})
****************************************************************************************************
LABEL ENCODING OF: appetite
```

```
contcols=set(data.dtypes[data.dtypes!='O'].index.values)# only fetech the float and int type columns
#contcols=pd.DataFrame(data,columns=contcols)
print(contcols)
```

```
{'blood_urea', 'serum_creatinine', 'albumin', 'blood_pressure', 'blood glucose random', 'sugar', 'sodium', 'hemoglobin', 'specific_gravit
y', 'age', 'potassium'}
```

Same as we did with categorical columns, we are majing use of **dtypes** for finding the continuous columns

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

If we observe the output of the above code we can observe that some columns have few values or you can say classes which can be considered as categorical columns. So , let's remove it and add the columns which we observed into their respective

So , let's remove it and add the columns which we observed into their respective variables.

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

With the help of add() function we can add an element.

```
contcols.add('red_blood_cell_count') # using add we can add the column
contcols.add('packed_cell_volume')
contcols.add('white_blood_cell_count')
print(contcols)
```

```
{'blood_urea', 'serum_creatinine', 'packed_cell_volume', 'blood_pressure', 'blood glucose random', 'sodium', 'hemoglobin', 'red_blood_cell
_count', 'age', 'potassium', 'white_blood_cell_count'}
```

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

```
{'hypertension', 'class', 'albumin', 'coronary_artery_disease', 'anemia', 'sugar', 'red_blood_cells', 'specific_gravity', 'bacteria', 'ped
al_edema', 'appetite', 'pus_cell', 'diabetesmellitus', 'pus_cell_clumps'}
```

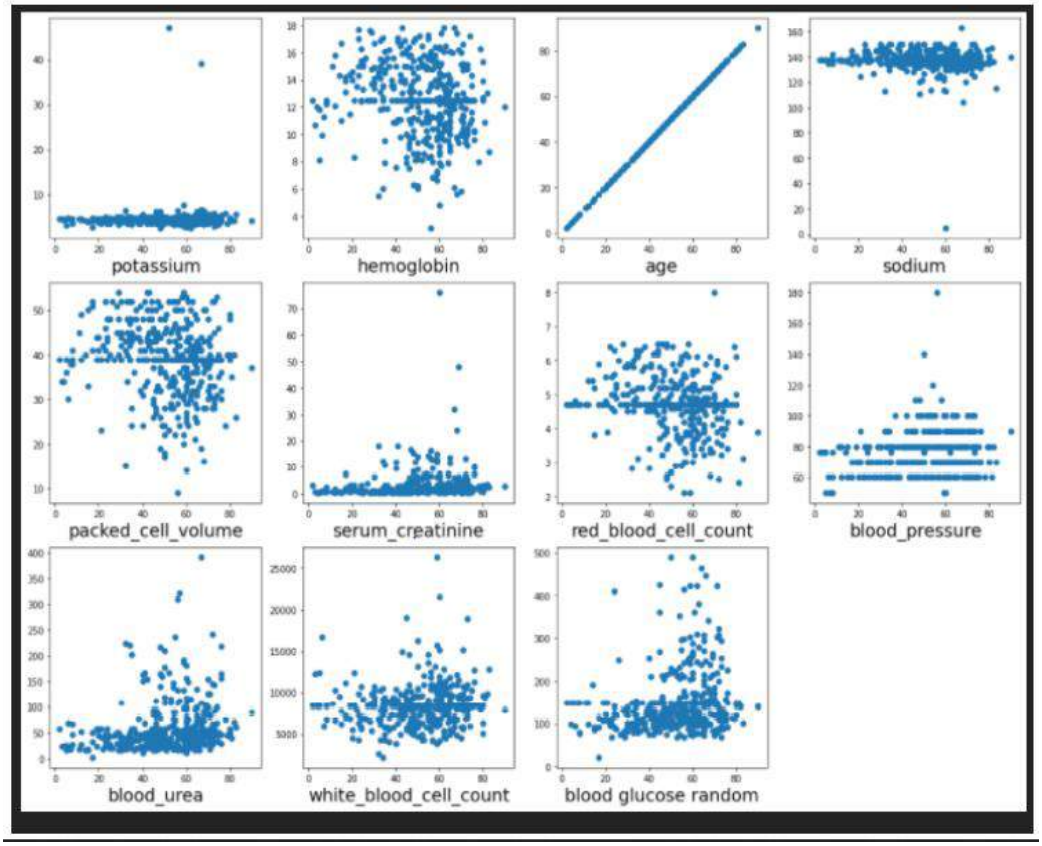
In our data some columns some unwanted classes so we have to rectify that also for

Age vs all continuous columns

Age vs all continous columns ¶

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

for column in contcols:
    if plotnumber<=11 :     # as there are 11 continous columns in the data
        ax = plt.subplot(3,4,plotnumber) # 3,4 is refer to 3X4 matrix
        plt.scatter(data['age'],data[column]) #plotting scatter plot
        plt.xlabel(column,fontsize=20)
        #plt.ylabel('Salary',fontsize=20)
    plotnumber+=1
plt.show()
```



Compare the model

```
from sklearn import model_selection
```

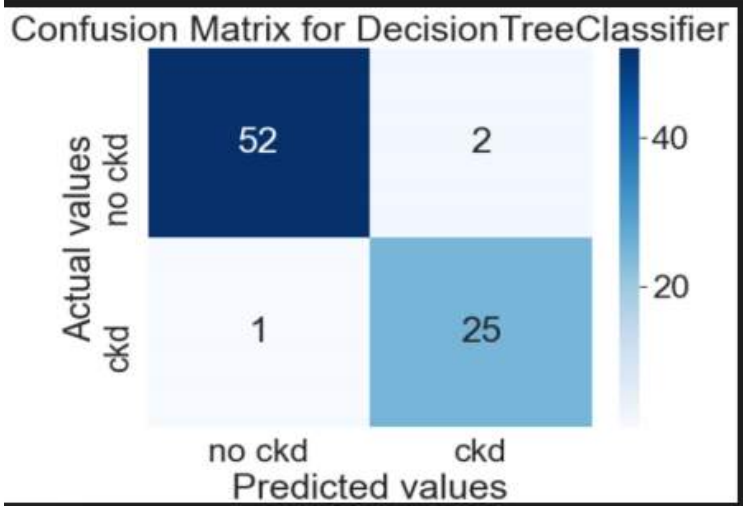
```
    dfs = []
    models = [
                ('LogReg', LogisticRegression()),
                ('RF', RandomForestClassifier()),
                ('DecisionTree',DecisionTreeClassifier()),

            ]
    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.KFold(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
```

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predict)
cm
```

```
array([[52,  2],
       [ 1, 25]], dtype=int64)
```

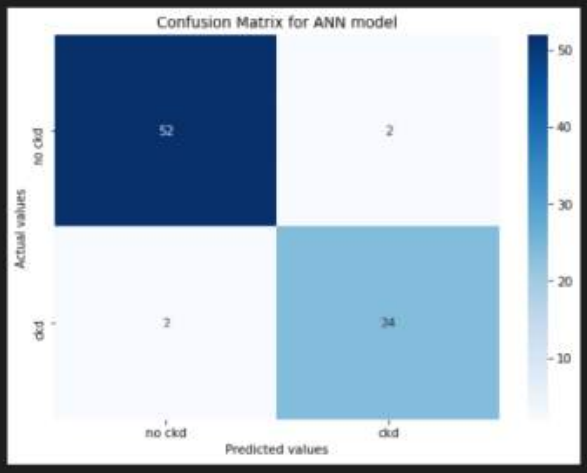
```python
# Plotting confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, cmap='Blues', annot=True, xticklabels=['no ckd', 'ckd'], yticklabels=['no ckd', 'ckd'])
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.title('Confusion Matrix for DecisionTreeClassifier')
plt.show()
```



```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[52,  2],
       [ 2, 24]], dtype=int64)
```

```python
# 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 ANN model')
plt.show()
```



```
bootstraps = []
for model in list(set(final.model.values)):
    model_df = final.loc[final.model == model]
    bootstrap = model_df.sample(n=30, replace=True)
    bootstraps.append(bootstrap)

bootstrap_df = pd.concat(bootstraps, ignore_index=True)
results_long = pd.melt(bootstrap_df,id_vars=['model'],var_name='metrics', value_name='values')
time_metrics = ['fit_time','score_time'] # fit time metrics
## PERFORMANCE METRICS
results_long_nofit = results_long.loc[~results_long['metrics'].isin(time_metrics)] # get df without fit data
results_long_nofit = results_long_nofit.sort_values(by='values')
## TIME METRICS
results_long_fit = results_long.loc[results_long['metrics'].isin(time_metrics)] # df with fit data
results_long_fit = results_long_fit.sort_values(by='values')
```

-

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and to be able to use it in the future.

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

```
data.isnull().any() #it will return true 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 random      True
blood_urea                True
serum_creatinine          True
sodium                    True
potassium                 True
hemoglobin                True
packed_cell_volume        True
white_blood_cell_count    True
red_blood_cell_count      True
hypertension              True
diabetesmellitus          True
coronary_artery_disease   True
appetite                  True
pedal_edema               True
anemia                    True
class                    False
dtype: bool
```

```
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 random      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  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
```