ADVANCED ACADEMIC CENTER

A CENTER FOR INTER-DISCIPLINARY RESEARCH

2020-2021

TITLE

“CHRONIC KIDNEY DISEASE PREDICTION”

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

Advanced Academic Center

(A Center For Inter-Disciplinary Research)

This is to certify that the project titled

"CHRONIC KIDNEY DISEASE PREDICTION"

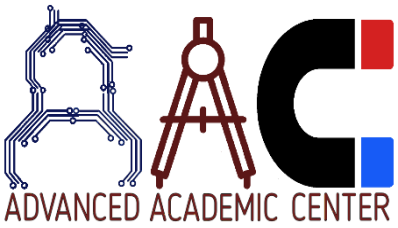
is a bonafide work carried out by the followingstudents in partial fulfilment of the requirements for Advanced Academic Center intern, submitted to the chair, AAC during the academic year 2020-2021.

NAME	ROLL NO.	BRANCH
LAKSHMI SAI NIHARIKA VULCHI	19241A04R4	ECE-E
GALIDEVARA SAI VENKATA MEGHANA	19241A04Q1	ECE-E
GOLLAPUDI VENKATA MAHATI	19241A04Q3	ECE-E

This work was not submitted or published earlier for any study.

Dr.B.R.K.Reddy
Program Coordinator

Dr.Ramamurthy Suri
Associate Dean, AAC



ACKNOWLEDGEMENTS

We express our deep sense of gratitude to our respected Director, Gokaraju Rangaraju Institute of Engineering and Technology, for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we extend our appreciation to our respected Principal, for permitting us to carry out this project.

We are thankful to the Associate Dean, Advanced Academic Centre, for providing us an appropriate environment required for the project completion. We are grateful to our project supervisor who spared valuable time to influence us with their novel insights.

We are indebted to all the above-mentioned people without whom we would not have concluded the project.

TABLE OF CONTENTS

ABSTRACT:

In the current era, everyone is trying to be conscious about health. But, due to the busy routines, one can take care of the health only if there are symptoms of any disorder or any such diagnosis is done. Chronic Kidney Disease (CKD) or chronic renal disease has become a major issue with a steady growth rate. A person can only survive without kidneys for an average time of 18 days, which makes a huge demand for a kidney transplant and dialysis. CKD doesn't show any symptoms or in some cases it doesn't show any disease specific symptoms. So, it is hard to predict, detect and prevent such a disease and this can lead to permanent health damage.

In this situation, machine learning can be used for prediction and analysis. Machine learning methods are efficient in CKD prediction. The CKD status is predicted based on clinical data, incorporating data pre-processing, a missing value handling method with collaborative filtering and attributes selection. We have used a dataset of attributes like age, blood pressure, serum creatinine, etc., to carry out the prediction. Algorithms like decision tree, k-means etc., have been used.

INTRODUCTION:

Chronic kidney disease (CKD) is a condition characterized by a gradual loss of kidney function over time. CKD is also known as chronic renal disease. Complications like high blood pressure, anaemia, weak bones, poor nutritional health, nerve damage, heart and blood vessel diseases may arise. These issues occur slowly over a long period of time.

Facts:

- Early detection can prevent the advancement of kidney disease to kidney failure.
- Glomerular filtration rate (GFR) is the best estimate of kidney function.
- Persistent proteinuria (protein in the urine) indicates the presence of CKD.
- African Americans, Hispanics, Pacific Islanders, American Indians and Seniors experience surged threat.
- Tests that can determine CKD are blood pressure, urine albumin and serum creatinine.

Common symptoms of CKD:

- Tiredness and low energy
- Poor appetite
- Insomnia
- Muscle cramping at night
- Swollen feet and ankles
- Puffiness under eyes
- Dry, itchy skin
- Polyuria

Other factors that increase the risk of CKD:

- Diabetes
- High blood pressure
- Inheritance

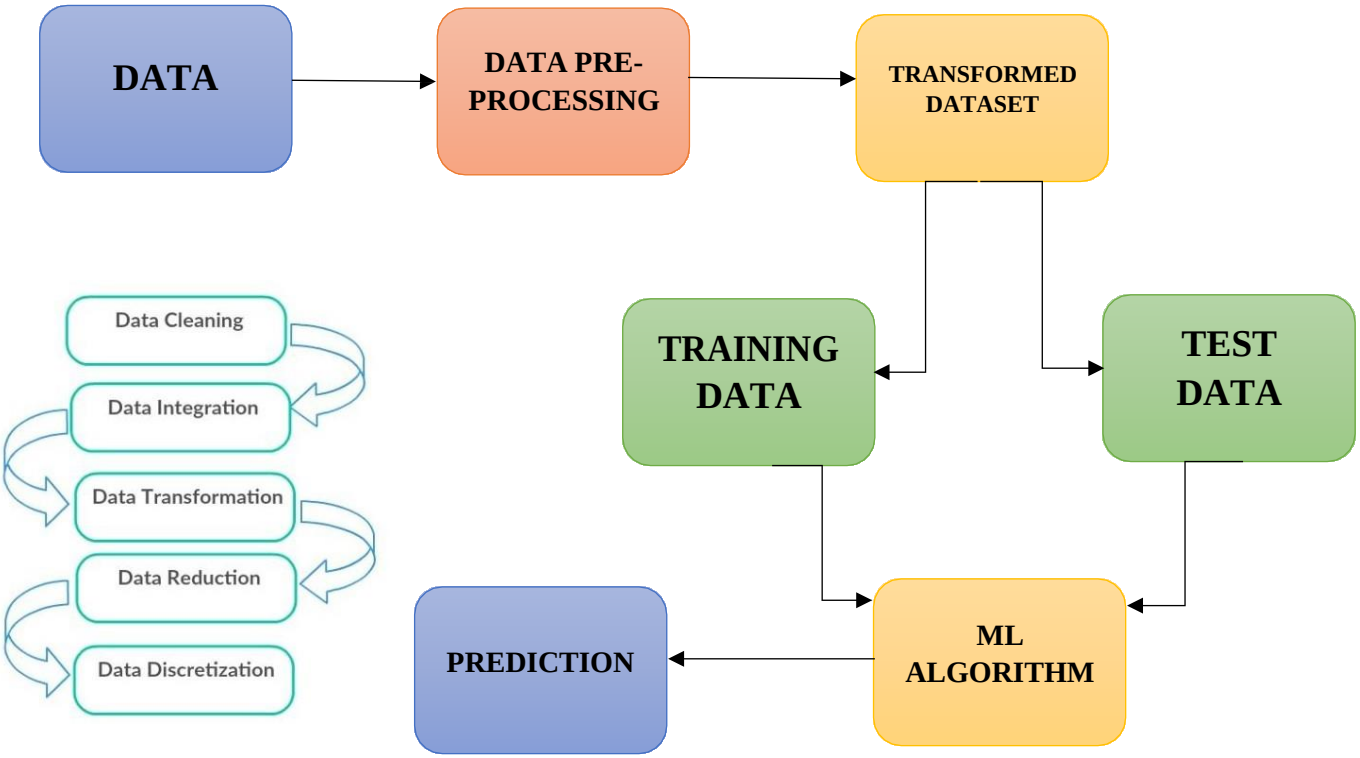
Based on the above seeings, CKD has got much importance for being prevented beforehand than treating after diagnosis. To hasten the cure of CKD, our ML model comes in handy.
The training dataset contains different attributes that are used to predict the probability of existence of CKD. Few of them are age, blood pressure(bp), specific gravity(sg) , platelet count(pc), red blood cells(rbc), serum creatinine(sc), sodium and potassium levels, packed cell volume(pcv), hypertension(htn), haemoglobin(hemo), coronary artery disease(cad), pulmonary embolism(pe), etc. We have implemented various ML algorithms to achieve this.

PROJECT WORKFLOW:

SOFTWARE USED:

Machine learning is an approach of data analysis that automates analytical model building. It is a subset of artificial intelligence which emphasizes that the machine can learn from data, identify patterns and make decisions with minimal human intervention.

HOW WE PROCEEDED:



To implement the algorithms, **python** programming language is used.

WHY PYTHON?

- Beginner friendly
- Most mature package libraries
- Versatile and Flexible
- Most popular in ML world

INCLUDED LIBRARIES:

1. **import pandas as pd**

Pandas:

- ✓ pandas is a software library written for python programming language for data manipulation and analysis.
- ✓ In particular, it offers data structures and operations for manipulating numerical tables and time series.
- ✓ It adds the missing piece to the SciPy framework for handling data.

2. import numpy as nm

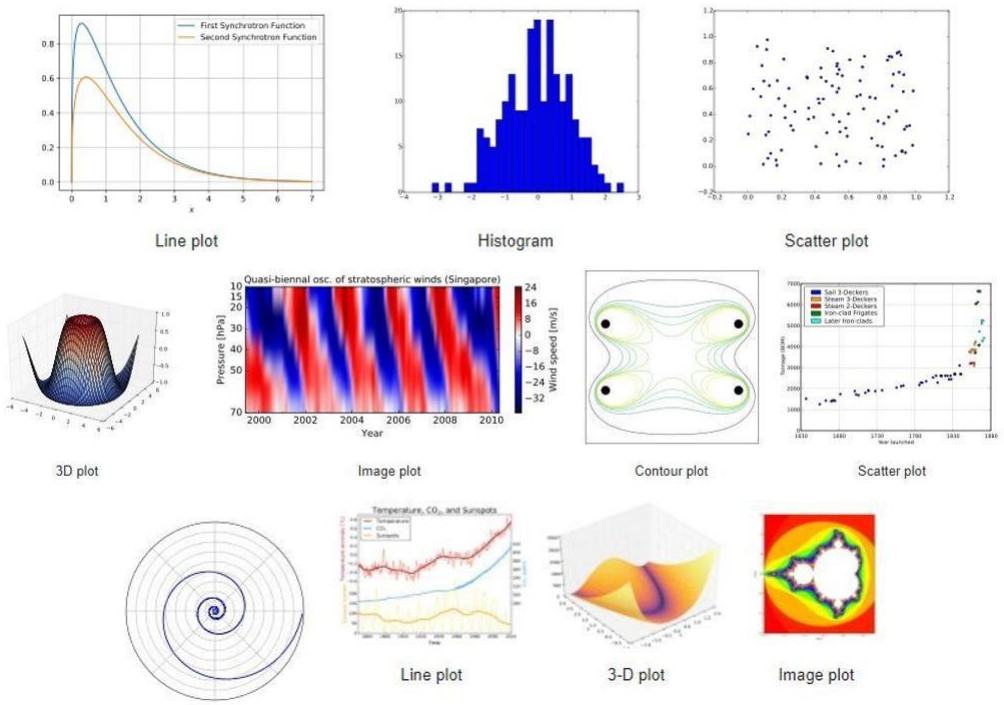
Numpy:

- ✓ numpy is a library for the python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- ✓ It also has functions for working in domain of linear algebra, Fourier transform and matrices.

3. import matplotlib.pyplot as mp

matplotlib.pyplot:

- ✓ It is a core object that contains the methods to create all sorts of charts and features in a plot.
- ✓ pyplot is a collection of command style functions that make matplotlib work like matlab.
- ✓ Each pyplot function makes some changes to a figure - e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.



sklearn

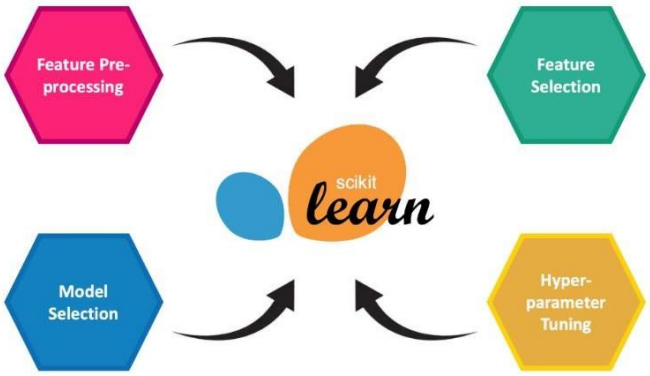
- ✓ Sci-kit learn is a free software machine learning library for the python programming language.
- ✓ It features various classification, regression and clustering algorithms including SVM, RFR, gradient boosting, K-means and DBSCAN.
- ✓ It is designed to interoperate with the python numerical and scientific libraries numpy and scipy.

4. from sklearn.impute import SimpleImputer

- ✓ The SimpleImputer class provides basic strategies for imputing missing values.
- ✓ Missing values can be imputed with a provided constant value, or using the statistics (mean, median or most frequent) of each column in which the missing values are located.
- ✓ This class also allows for different missing values encodings.

5. from sklearn.preprocessing import Imputer

- ✓ The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.
- ✓ In general, learning algorithms benefit from standardization of the dataset.



Imputer - It is good practice to identify and replace missing values for each column in your input data prior to modelling your prediction task. This is called missing data imputation, or imputing for short.

6. from sklearn.preprocessing import LabelEncoder, OneHotEncoder

One Hot Encoding –

- ✓ One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction.
- ✓ With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns.
- ✓ Each integer value is represented as a binary vector.

LabelEncoder –

- ✓ Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form.
- ✓ Machine learning algorithms can then decide in a better way on how those labels must be operated.
- ✓ It is an important pre-processing step for the structured dataset in supervised learning.

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

7. from sklearn.compose import ColumnTransformer

sklearn.compose-

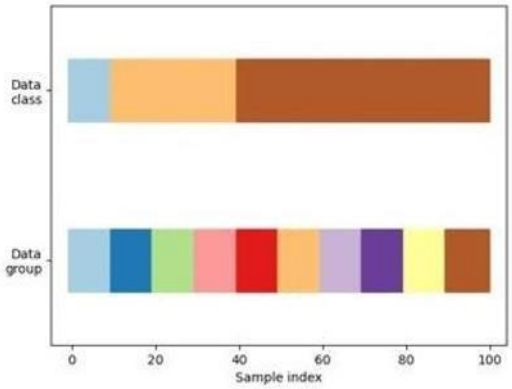
- ✓ It applies transformers to columns of an array or pandas DataFrame.
- ✓ This estimator allows different columns of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space.

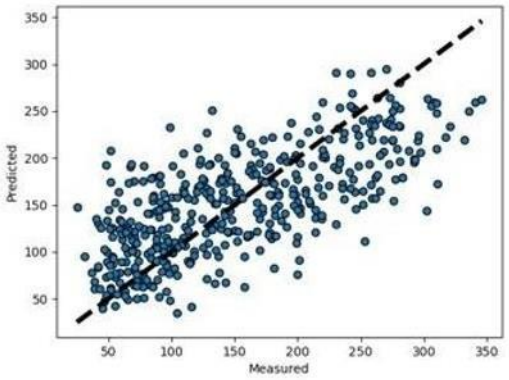
ColumnTransformer-

- ✓ The ColumnTransformer is a class in the scikit-learn Python machine learning library that allows you to selectively apply data preparation transforms.

8. from sklearn.model_selection import train_test_split

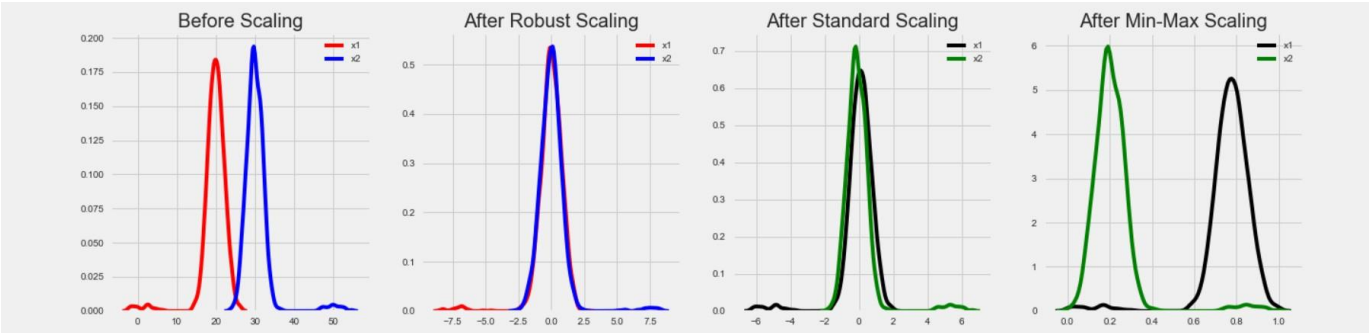
- ✓ model selection is a method for setting a blueprint to analyse data and then using it to measure new data.
- ✓ Selecting a proper model allows you to generate accurate results when making a prediction.
- ✓ To do that, you need to train your model by using a specific dataset and test it against another dataset.
- ✓ Model selection is a process that be applies across different types of models.





9. from sklearn.preprocessing import StandardScaler

- ✓ StandardScaler follows Standard Normal Distribution (SND).
- ✓ Therefore, it makes mean = 0 and scales the data to unit variance.
- ✓ This method removes the median and scales the data in the range between 1st quartile and 3rd quartile.

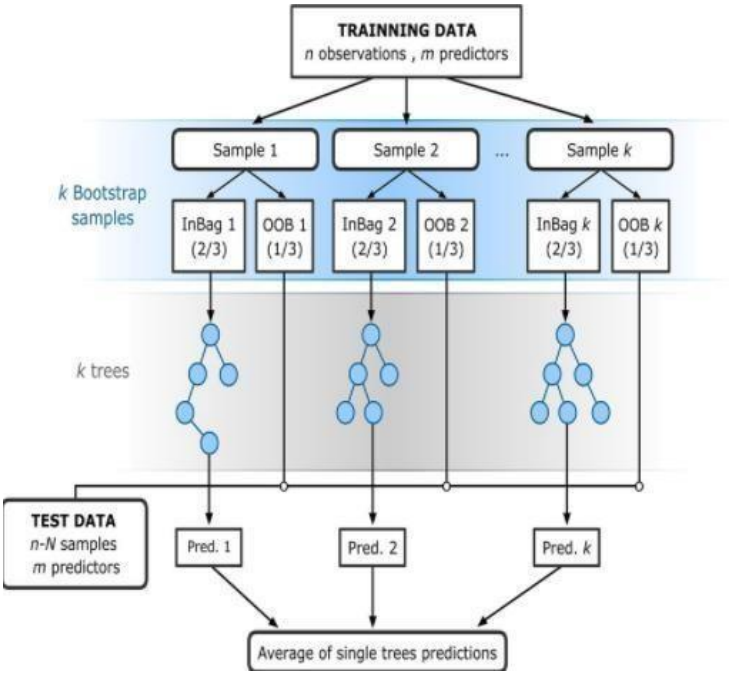


ALGORITHMS APPLIED:

i. Random Forest Regression

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees. We need to approach the Random Forest regression technique like any other machine learning technique:

• Design a specific question or data and get the source to determine the required data.
• Make sure the data is in an accessible format else convert it to the required format.
• Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
• Create a machine learning model
• Set the baseline model that you want to achieve
• Train the data machine learning model.
• Provide an insight into the model with test data
• Now compare the performance metrics of both the test data and the predicted data from the model.
• If it doesn't satisfy your expectations, you can try improving your model accordingly or dating your data or use another data modelling technique.
• At this stage you interpret the data you have gained and report accordingly

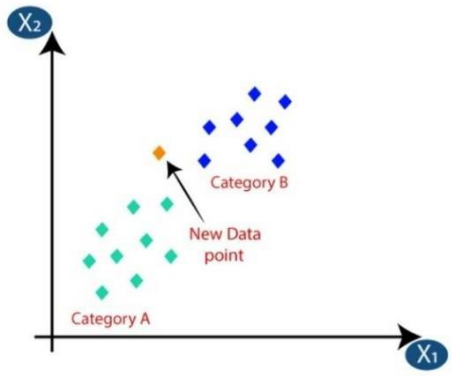


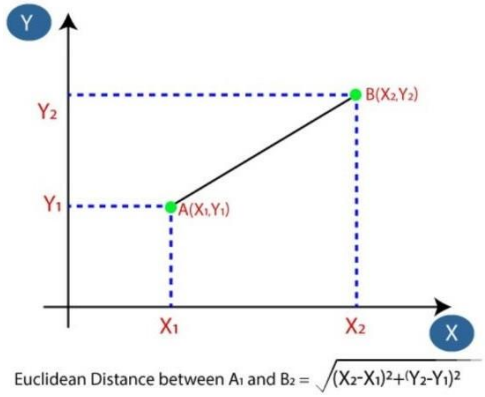
ii. K-Nearest Neighbor (KNN)

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity
- This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

HOW DOES KNN WORK?

Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of K number of neighbors
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
Step-6: Our model is ready.





How to select the value of K in the K-NN Algorithm?

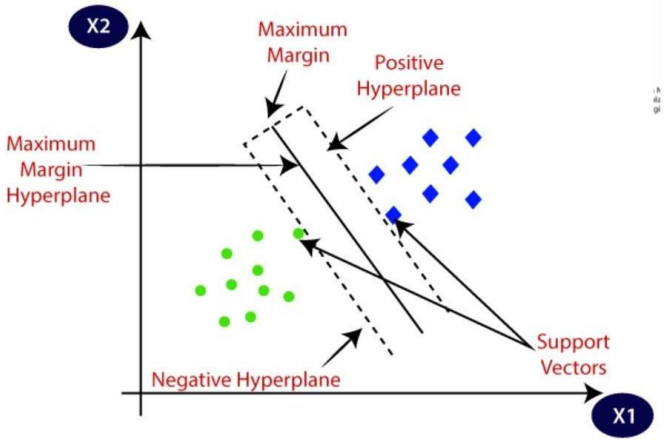
Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them.
- The most preferred value for K is 5. A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.

- Large values for K are good, but it may find some difficulties.

iii. Single Vector Machine (SVM)

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems
- However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
- This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

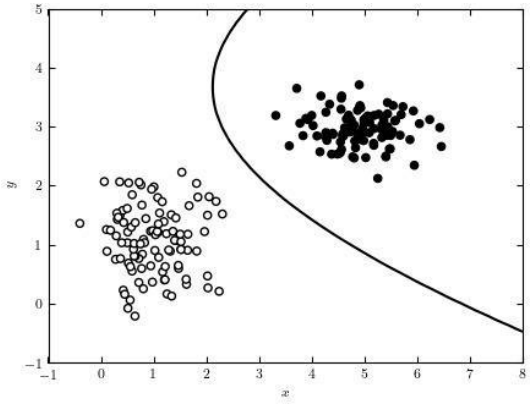


iv. Naive Bayes Classifier

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

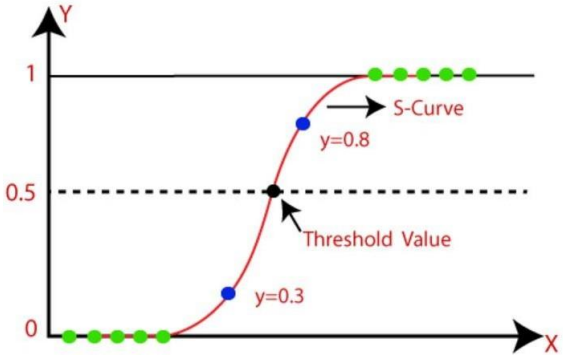
Gaussian model of Naive Bayes:

The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.



v. Logistic Regression

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique.
- It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable.
- Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.



Assumptions for Logistic Regression:

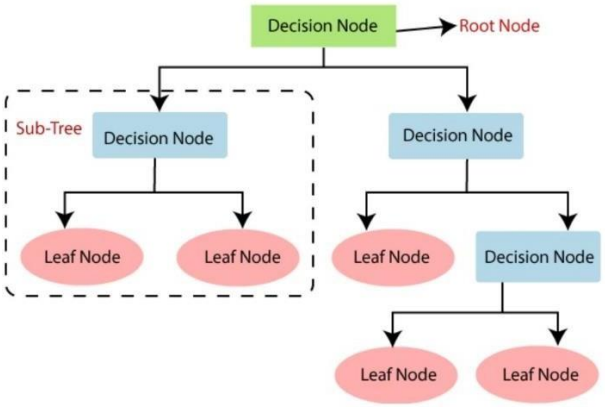
The dependent variable must be categorical in nature. The independent variable should not have multi-collinearity.

Logistic Regression Equation:

y=b0+b1x1+b2x2+ .....+bnxn

vi. Decision Tree Classifier

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node.
- Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.



HOW DOES THIS ALGORTIHM WORK?

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contains possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

IMPLEMENTATION:

Code:

```
import pandas as pd
import numpy as nm
import matplotlib.pyplot as mp
ds=pd.read_csv('updated data.csv')
print(ds)
#nv stands for numerical values and cv for categorical values
nv=ds.columns[ds.dtypes != 'object']
cv=ds.columns[ds.dtypes == 'object']
print(nv)
print(cv)
#identifying the number of Nan values for each column using the isnull() function
ds.isnull().sum()
#finding the column numbers for each numerical data values' columns
ds.columns.get_indexer(['age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot',
      'hemo', 'pcv', 'wc', 'rc'])
#finding the column numbers for each categorical data values' columns
ds.columns.get_indexer(['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane'])
#importing the simple imputer class and replacing the nan values for numerical data
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import Imputer

imr = Imputer(missing_values='NaN', strategy='median', axis=0)
imr = imr.fit(ds[['age']])
ds['age'] = imr.transform(ds[['age']]).ravel()
imr = imr.fit(ds[['bp']])
ds['bp'] = imr.transform(ds[['bp']]).ravel()
imr = imr.fit(ds[['sg']])
ds['sg'] = imr.transform(ds[['sg']]).ravel()
imr = imr.fit(ds[['al']])
ds['al'] = imr.transform(ds[['al']]).ravel()
imr = imr.fit(ds[['su']])
ds['su'] = imr.transform(ds[['su']]).ravel()
```

```
imr = imr.fit(ds[['bgr']])
ds['bgr'] = imr.transform(ds[['bgr']]).ravel()
imr = imr.fit(ds[['bu']])
ds['bu'] = imr.transform(ds[['bu']]).ravel()
imr = imr.fit(ds[['sc']])
ds['sc'] = imr.transform(ds[['sc']]).ravel()
imr = imr.fit(ds[['sod']])
ds['sod'] = imr.transform(ds[['sod']]).ravel()
imr = imr.fit(ds[['pot']])
ds['pot'] = imr.transform(ds[['pot']]).ravel()
imr = imr.fit(ds[['hemo']])
ds['hemo'] = imr.transform(ds[['hemo']]).ravel()
imr = imr.fit(ds[['pcv']])
ds['pcv'] = imr.transform(ds[['pcv']]).ravel()
imr = imr.fit(ds[['wc']])
ds['wc'] = imr.transform(ds[['wc']]).ravel()
imr = imr.fit(ds[['rc']])
ds['rc'] = imr.transform(ds[['rc']]).ravel()
#True represents a Nan value in the column and we replace it with the mode.We calculate mo
de for each column using the mode() function
ds['rbc'].fillna('normal',inplace =True)
ds['pc'].fillna('normal' , inplace= True)
ds['pcc'].fillna('notpresent' , inplace= True)
ds['ba'].fillna('notpresent' , inplace= True)
ds['htn'].fillna('no' , inplace= True)
ds['dm'].fillna('no' , inplace= True)
ds['cad'].fillna('no' , inplace= True)
ds['appet'].fillna('good' , inplace= True)
ds['pe'].fillna('no' , inplace= True)
ds['ane'].fillna('no' , inplace= True)
#we check the data set if all the Nan values are replaced
print(ds)
ds['classification']=ds['classification'].replace('ckd\t','ckd')
#extracting the x and y values awhere x represents the independent columns(input columns) a
nd y represents the dependent column(ouput column)
x=ds.iloc[:,:-1].values
y=ds.iloc[:,-1].values
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn.compose import ColumnTransformer
columnTransformer = ColumnTransformer([('encoder',OneHotEncoder(),[5,  6,  7,  8, 18, 19,
20, 21, 22, 23])] ,remainder='passthrough')
x= columnTransformer.fit_transform(x)
lb_y=LabelEncoder()
y=lb_y.fit_transform(y)
#parameters are test_size and random_state.test_size decides the size of testing data in the im
put columns and random_state
#shuffles the rows in our data set
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.8, random_state=2)
#feature scaling is performed using the StandardScaler
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
#n_components is the input parameter to PCA.Initially we give it as "none" and find the expl
ained variance of all the columns
from sklearn.decomposition import PCA
pca = PCA(n_components=None)
x = pca.fit_transform(x)
var = pca.explained_variance_ratio_
print(var)
#after we know the explained variance of all the columns we give the n_components value ba
sed upon the number of values with a greater variance
from sklearn.decomposition import PCA
pca = PCA(n_components=9)
x = pca.fit_transform(x)
var = pca.explained_variance_ratio_
print(var)
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn_cla = KNeighborsClassifier(n_neighbors=7,metric='minkowski',p=2)
knn_cla.fit(x_train,y_train)

knn_pred = knn_cla.predict(x_test)
from sklearn.metrics import accuracy_score,confusion_matrix
acc_knn = accuracy_score(y_test,knn_pred)
cm_knn = confusion_matrix(y_test,knn_pred)
print(acc_knn)
print(cm_knn)
precision_knn=158/(158+39)
recall_knn=158/(158+0)
f_knn=(2*precision_knn*recall_knn)/(precision_knn+recall_knn)
print(precision_knn)
print(recall_knn)
print(f_knn)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
print(mean_absolute_error(y_test,knn_pred))
print(mean_squared_error(y_test,knn_pred))
print(r2_score(y_test,knn_pred))
print(nm.sqrt(mean_squared_error(y_test,knn_pred)))
```

DECISION TREE

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(x_train,y_train)
```

```
dt_pred=regressor.predict(x_test)
print(dt_pred)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
print(mean_absolute_error(y_test,dt_pred))
print(mean_squared_error(y_test,dt_pred))
print(r2_score(y_test,dt_pred))
print(nm.sqrt(mean_squared_error(y_test,dt_pred)))
acc_dt = accuracy_score(y_test,dt_pred)
cm_dt = confusion_matrix(y_test,dt_pred)
print(acc_dt)
print(cm_dt)
precision_dt=187/(187+10)
recall_dt=187/(187+14)
f_dt=(2*precision_dt*recall_dt)/(precision_dt+recall_dt)
print(precision_dt)
print(recall_dt)
print(f_dt)
```

RANDOM FOREST

```
# creating and training our randome forest algorithm
from sklearn.ensemble import RandomForestRegressor
# n_estimators argument asks the number of decision tress you want the RF algorithm to use.
rfr = RandomForestRegressor(n_estimators=80,random_state=0)
rfr.fit(x_train,y_train)
```

```
rfr_pred=rfr.predict(x_test)
print(rfr_pred)
print(mean_absolute_error(y_test,rfr_pred))
print(mean_squared_error(y_test,rfr_pred))
print(r2_score(y_test,rfr_pred))
print(nm.sqrt(mean_squared_error(y_test,rfr_pred)))
```

LOGISTIC

```
from sklearn.linear_model import LogisticRegression
logisticreg = LogisticRegression()
logisticreg.fit(x_train, y_train)
lr_pr = logisticreg.predict(x_test)
```

```
com_lr = confusion_matrix(y_test,lr_pr)
print(com_lr)
accuracy_lr= accuracy_score(y_test,lr_pr)
print(accuracy_lr)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
print(mean_absolute_error(y_test,lr_pr))
print(mean_squared_error(y_test,lr_pr))
print(r2_score(y_test,lr_pr))
print(nm.sqrt(mean_squared_error(y_test,lr_pr)))
precision_lr=177/(177+20)
recall_lr=177/(177+0)
```

```
f_lr=(2*precision_lr*recall_lr)/(precision_lr+recall_lr)
print(precision_lr)
print(recall_lr)
print(f_lr)
```

naive bayes

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
nb_pn=model.predict(x_test)
print(nb_pn)

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,nb_pn))
comn_nb= confusion_matrix(y_test,nb_pn)
print(comn_nb)
print(mean_absolute_error(y_test,lr_pr))
print(mean_squared_error(y_test,lr_pr))
print(r2_score(y_test,lr_pr))
print(nm.sqrt(mean_squared_error(y_test,lr_pr)))
precision_nb=188/(188+9)
recall_nb=188/(188+0)
f_nb=(2*precision_nb*recall_nb)/(precision_nb+recall_nb)
print(precision_nb)
print(recall_nb)
print(f_nb)
```

SVM

```
from sklearn.svm import SVC
classifier = SVC(kernel='linear',random_state=41)
classifier.fit(x_train,y_train)
ysvm = classifier.predict(x_test)
print(ysvm)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
print(mean_absolute_error(y_test,ysvm))
print(mean_squared_error(y_test,ysvm))
print(r2_score(y_test,ysvm))
print(nm.sqrt(mean_squared_error(y_test,ysvm)))
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,ysvm))
coms= confusion_matrix(y_test,ysvm)
print(coms)
prec_svm=180/(180+17)
recall_svm=180/(180+0)
f_svm=(2*prec_svm*recall_svm)/(prec_svm+recall_svm)
```

```
print(prec_svm,recall_svm,f_svm)
```

VOTING CLASSIFIERS

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier,RandomForestClassifier,VotingClassifier
bag=BaggingClassifier(RandomForestClassifier(n_estimators=80,criterion='entropy',random
_state=0),max_samples=1.0,max_features=1.0,n_estimators=10,random_state=0)
bag.fit(x_train,y_train)
bag_pred=bag.predict(x_test)
print(confusion_matrix(y_test,bag_pred))
print(accuracy_score(y_test,bag_pred))
```

```
prec_bag=194/(194+3)
recall_bag=194/(194+1)
f_bag=(2*prec_bag*recall_bag)/(prec_bag+recall_bag)
print(prec_bag,recall_bag,f_bag)
```

FUTURE DEVELOPMENTS:

We are planning to implement our model onto a mobile application which offers the following features:

- The reports of the patient can be viewed according to the inputs given, by making use of our ML model.
- Diet plan, essential medication and precautions or suggestions are also included.
- If the patient's reports are abnormal, he/she is suggested to visit a nephrologist.

CONCLUSION:

In this project we have worked on different machine learning algorithms. We have analysed 26 different attributes that affect the CKD patients and predicted the accuracy. From the prediction analysis, it is noted that the decision tree algorithm gives the accuracy of 94.9%, SVM gives accuracy of 91.3%, voting classifier gives 98.4% accuracy. Our model will help the doctors to treat the patients early and also diagnose more patients with the disease within a short span.

REFERENCES:

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5414519/
- https://www.researchgate.net/publication/341398109_Chronic_Kidney_Disease_Prediction_using_machine_Learning_Models
  https://www.thelancet.com/action/showFullTableHTML?isHtml=true&tableId=tbl2&pii=S0140-6736%2820%2930045-3
- https://www.kaggle.com/mansoordaku/ckdisease/activity
- https://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease#
- https://towardsdatascience.com/how-voting-classifiers-work-f1c8e41d30ff
- https://github.com/liuy14/Kidney_Disease_Detection/tree/master/Kidney%20Disease%20Study
- https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/