**SAMPLE CODINGS:**

# Importing Libraries:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# for displaying all feature from dataset:

pd.pandas.set\_option('display.max\_columns', None)

# Reading Dataset:

dataset = pd.read\_csv("Kidney\_data.csv")

# Top 5 records:

dataset.head()

# Dropping unneccsary feature :

dataset = dataset.drop('id', axis=1)

# Shape of dataset:

dataset.shape

# Cheaking Missing (NaN) Values:

dataset.isnull().sum()

# Description:

dataset.describe()

# Datatypes:

dataset.dtypes

dataset.head()

dataset['rbc'].value\_counts()

dataset['rbc'] = dataset['rbc'].replace(to\_replace = {'normal' : 0, 'abnormal' : 1})

dataset['pc'].value\_counts()

dataset['pc'] = dataset['pc'].replace(to\_replace = {'normal' : 0, 'abnormal' : 1})

dataset['pcc'].value\_counts()

dataset['pcc'] = dataset['pcc'].replace(to\_replace = {'notpresent':0,'present':1})

dataset['ba'].value\_counts()

dataset['ba'] = dataset['ba'].replace(to\_replace = {'notpresent':0,'present':1})

dataset['htn'].value\_counts()

dataset['htn'] = dataset['htn'].replace(to\_replace = {'yes' : 1, 'no' : 0})

dataset['dm'].value\_counts()

dataset['dm'] = dataset['dm'].replace(to\_replace = {'\tyes':'yes', ' yes':'yes', '\tno':'no'})

dataset['dm'] = dataset['dm'].replace(to\_replace = {'yes' : 1, 'no' : 0})

dataset['cad'].value\_counts()

dataset['cad'] = dataset['cad'].replace(to\_replace = {'\tno':'no'})

dataset['cad'] = dataset['cad'].replace(to\_replace = {'yes' : 1, 'no' : 0})

dataset['appet'].unique()

dataset['appet'] = dataset['appet'].replace(to\_replace={'good':1,'poor':0,'no':np.nan})

dataset['pe'].value\_counts()

dataset['pe'] = dataset['pe'].replace(to\_replace = {'yes' : 1, 'no' : 0})

dataset['ane'].value\_counts()

dataset['ane'] = dataset['ane'].replace(to\_replace = {'yes' : 1, 'no' : 0})

dataset['classification'].value\_counts()

dataset['classification'] = dataset['classification'].replace(to\_replace={'ckd\t':'ckd'})

dataset["classification"] = [1 if i == "ckd" else 0 for i in dataset["classification"]]

dataset.head()

# Datatypes:

dataset.dtypes

dataset['pcv'] = pd.to\_numeric(dataset['pcv'], errors='coerce')

dataset['wc'] = pd.to\_numeric(dataset['wc'], errors='coerce')

dataset['rc'] = pd.to\_numeric(dataset['rc'], errors='coerce')

# Datatypes:

dataset.dtypes

# Description:

dataset.describe()

# Cheaking Missing (NaN) Values:

dataset.isnull().sum().sort\_values(ascending=False)

dataset.columns

features = ['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',

           'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',

           'appet', 'pe', 'ane']

for feature in features:

    dataset[feature] = dataset[feature].fillna(dataset[feature].median())

dataset.isnull().any().sum()

plt.figure(figsize=(24,14))

sns.heatmap(dataset.corr(), annot=True, cmap='YlGnBu')

plt.show()

dataset.drop('pcv', axis=1, inplace=True)

dataset.head()

# Target feature:

sns.countplot(dataset['classification'])

# Independent and Dependent Feature:

X = dataset.iloc[:, :-1]

y = dataset.iloc[:, -1]

X.head()

# Feature Importance:

from sklearn.ensemble import ExtraTreesClassifier

import matplotlib.pyplot as plt

model=ExtraTreesClassifier()

model.fit(X,y)

plt.figure(figsize=(8,6))

ranked\_features=pd.Series(model.feature\_importances\_,index=X.columns)

ranked\_features.nlargest(24).plot(kind='barh')

plt.show()

ranked\_features.nlargest(8).index

X = dataset[['sg', 'htn', 'hemo', 'dm', 'al', 'appet', 'rc', 'pc']]

X.head()

X.tail()

y.head()

# Train Test Split:

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size=0.3, random\_state=33)

print(X\_train.shape)

print(X\_test.shape)

# Importing Performance Metrics:

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# RandomForestClassifier:

from sklearn.ensemble import RandomForestClassifier

RandomForest = RandomForestClassifier()

RandomForest = RandomForest.fit(X\_train,y\_train)

# Predictions:

y\_pred = RandomForest.predict(X\_test)

# Performance:

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

# AdaBoostClassifier:

from sklearn.ensemble import AdaBoostClassifier

AdaBoost = AdaBoostClassifier()

AdaBoost = AdaBoost.fit(X\_train,y\_train)

# Predictions:

y\_pred = AdaBoost.predict(X\_test)

# Performance:

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

# GradientBoostingClassifier:

from sklearn.ensemble import GradientBoostingClassifier

GradientBoost = GradientBoostingClassifier()

GradientBoost = GradientBoost.fit(X\_train,y\_train)

# Predictions:

y\_pred = GradientBoost.predict(X\_test)

# Performance:

print('Accuracy:', accuracy\_score(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))