## Problem Definition

**CHAPTER-1 INTRODUCTION**

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 and we make use of such information to build a machine learning model that predicts Chronic Kidney Disease.

## Project Overview

The prediction of the model is classification type,  we apply a logistic regression algorithm on our dataset. There are many features that are responsible for Chronic Kidney Disease, e.g. age, blood pressure, sugar etc. For better prediction the Chronic kidney Disease, we should consider as many relevant features as possible. There are many researchers who work on prediction of CKD with the help of many different classification algorithm .The novel approach to detect CKD using machine learning algorithm is, We get result on dataset which having 400 records and 25 attributes which gives result of patient having CKD or not CKD. using k-nearest neighbours, random forest and neural network will get results.

* 1. **Hardware Specification** RAM: 16GB and Higher Processor: Intel i5 and above Hard Disk: 1000GB: Minimum

## Software Specification

OS : Windows

Python IDE : Anaconda python 2.7.x

## Existing System

**CHAPTER-2 LITERATURE SURVEY**

Chronic Kidney Disease (CKD) should be diagnosed earlier before kidneys fail to work. To help doctors or medical experts in prediction of CKD among patients easily, this paper has developed an expert system named Chronic Kidney Disease Prediction System (CKDPS) that can predict CKD among patients. The dataset used to develop CKDPS is taken from the Kaggle machine learning database. Before the implementation of CKDPS, different machine learning algorithms such as, k-Nearest Neighbors (KNN), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) algorithm are applied on the dataset and their performances are compared to the matter of accuracy, precision and recall results. Finally, Random Forest algorithm is chosen to implement CKDPS as it gives 100% accuracy, precision and recall results. This paper also compares the accuracy results of different machine learning algorithms from different previous related works where same or different CKD dataset has been used.

### Disadvantages of Existing System

* + - When data sizes increases, computational process becomes a challenging thing.
    - Limited number of combinations.
    - Low accuracy

## Proposed System

The step-by-step procedures for the proposed methodology adopted in this research are as follows :

**Collecting Data Sets**: ML depends heavily on data, it is the most crucial aspect tha makes algorithm training possible. Data sets can be collected from sources like kagge.com, data.gov etc.

**Cleaning the Data Sets:** Gather open source raw data of CKD patients available on internet. Data obtained from internet does not contains the name of the attribute so first we assigned the names to the attribute. Missing values in the dataset like NA’s or blank values are removed by using WEKA function “ReplaceMissingValues” used, which replaces NA’s with the mean values of that attribute.

**Data Categorization**: Supervised machine learning techniques used to take labelled data, which already have categorized in to the available classes. That’s why we need to assign label such as “Dependent” or “Independent” according to their characteristics.

**Training and Testing Data Sets:** The dataset is divided into two sub datasets both containing 14 attributes.

Training data: training dataset is derived from main dataset and it contains 300 out of 400 records in main dataset of CKD.

Testing data: testing dataset is of 100 out of 400 records from main CKD dataset.

**Prediction**:

Prediction using Decision tree:

Prediction done by Decision tree model trained by training CKD dataset .

**Prediction using SVM:**

Prediction done by SVM model trained by training CKD dataset

### Advantages

* + - We can create a good number of combinations.
    - This will increase the prediction accuracy of the matches.
    - Python has good graphical libraries.

The result can be project in a more effective way using graphical libraries in Python.

# CHAPTER-3

**SYSTEM ANALYSIS & DESIGN**

## Requirement Specification Software Requirements

OS : Windows

Python IDE : python 2.7.x and above Pycharm IDE

Setup tools and pip to be installed for 3.6.x

**Hardware Requirements** RAM: 16GB and Higher Processor: Intel i5 and above Hard Disk: 1000GB: Minimum **Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

## Design and Architecture

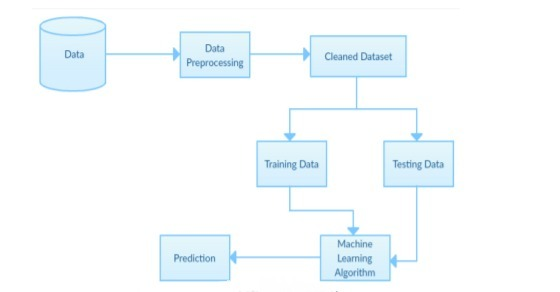


Fig. Architecture of Chronic Kidney Disease Analysis Using Machine Learning

* 1. **Algorithms and Sample Code Algorithms**

**Logistic regression:**

Logistic regression is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that in its basic form uses a [logistic](https://en.wikipedia.org/wiki/Logistic_function) function to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), although many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), **logistic regression** (or **logit regression**) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (a form of [binary regression](https://en.wikipedia.org/wiki/Binary_regression)).

Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labelled "0" and "1". In the logistic model, the [log-odds](https://en.wikipedia.org/wiki/Log-odds) (the [logarithm](https://en.wikipedia.org/wiki/Logarithm) of the [odds](https://en.wikipedia.org/wiki/Odds)) for the value labelled "1" is a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of one or more [independent variables](https://en.wikipedia.org/wiki/Independent_variable) ("predictors")

## Decision tree:

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.Decision trees can also be seen as generative models of induction rules from empirical data. An optimal decision tree is then defined as a tree that accounts for most of the data, while minimizing the number of levels.Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.

## Sample code

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=pd.read\_csv("chronickidneydisease.csv")

data.head()

data.tail()

data.head(10)

data.drop(["id"],axis=1,inplace=True) # drop is used for dropping the column

data.columns #return all the column names

data.columns=['age','blood\_pressure','specific\_gravity','albumin',

'sugar','red\_blood\_cells','pus\_cell','pus\_cell\_clumps','bacteria',

'blood glucose random','blood\_urea','serum\_creatinine','sodium','potassium',

'hemoglobin','packed\_cell\_volume','white\_blood\_cell\_count','red\_blood\_cell\_count',

'hypertension','diabetesmellitus','coronary\_artery\_disease','appetite',

'pedal\_edema','anemia','class'] # manually giving the name of the columns

data.columns

data.info()

data.describe()

data['class'].unique()

data['class']=data['class'].replace("ckd\t","ckd") #replace is used for renaming

data['class'].unique()

np.unique(data.dtypes,return\_counts=True)

catcols=set(data.dtypes[data.dtypes=='O'].index.values) # only fetch the object type columns

print(catcols)

for i in catcols:

print("Columns :",i)

print(c(data[i])) #using counter for checking the number of classess in the column

print('\*'\*120+'\n')

catcols.remove('red\_blood\_cell\_count') # remove is used for removing a particular column

catcols.remove('packed\_cell\_volume')

catcols.remove('white\_blood\_cell\_count')

print(catcols)

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)

for i in contcols:

print("Continous Columns :",i)

print(c(data[i]))

print('\*'\*120+'\n')

contcols.remove('specific\_gravity')

contcols.remove('albumin')

contcols.remove('sugar')

print(contcols)

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)

catcols.add('specific\_gravity')

catcols.add('albumin')

catcols.add('sugar')

print(catcols)

data['coronary\_artery\_disease'] = data.coronary\_artery\_disease.replace('\tno','no') # replacing \tno with no

c(data['coronary\_artery\_disease'])

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

c(data['diabetesmellitus'])

data.isnull().any() #it will return true if any columns is having null values

data.isnull().sum() #returns the count of null values present in each column

catcols=['anemia','pedal\_edema','appetite','bacteria','class','coronary\_artery\_disease','diabetesmellitus',

'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)

import matplotlib.pyplot as plt # import the matplotlib libaray

fig=plt.figure(figsize=(5,5)) #plot size

plt.scatter(data['age'],data['blood\_pressure'],color='blue')

plt.xlabel('age') #set the label for x-axis

plt.ylabel('blood pressure') #set the label for y-axis

plt.title("age VS blood Scatter Plot")

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()

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()

sns.countplot(data['class'])

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)

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)#train test split

print(x\_train.shape)

print(y\_train.shape)

print(x\_test.shape)

print(y\_test.shape)

from sklearn.linear\_model import LogisticRegression

lgr = LogisticRegression()

lgr.fit(x\_train,y\_train)

y\_pred = lgr.predict(x\_test)

y\_pred1 = lgr.predict([[129,99,1,0,0,1,0,1]])

print(y\_pred1)

c(y\_pred1)

accuracy\_score(y\_test,y\_pred)

conf\_mat = confusion\_matrix(y\_test,y\_pred)

conf\_mat

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

## HTML CODE:

### HOME PAGE:

<form action="/Prediction" method="[POST,GET]">

<header>

<div class="wrapper">

<div class="logo">

<img src="static/logo.ico" class="w3-round" alt="Norway">

</div>

<link rel="stylesheet" type="text/css" href="style.css">

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css') }}">

<ul class="nav-area">

<li><a href="#">Home</a></li>

<input type="submit" value="Prediction">

</ul>

</div>

<div class="welcome-text">

<h1>CHRONIC KIDNEY DISEASE PREDICTION</h1>

</div>

</header>

</form>

### INDEX PAGE:

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title>CKD Predictor</title>

<link rel="shortcut icon" href="{{ url\_for('static', filename='diabetes-favicon.ico') }}">

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='styles.css') }}">

<script src="https://kit.fontawesome.com/5f3f547070.js" crossorigin="anonymous"></script>

<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">

</head>

<body>

<!-- Website Title -->

<div class="container">

<h2 class='container-heading'><span class="heading\_font">Chronic Kidney Disease</span></h2>

<div class='description'>

<p>A Machine Learning Web App, Built with Flask</p>

</div>

</div>

<!-- Text Area -->

<div class="ml-container">

<form action="{{ url\_for('predict') }}" method="POST">

<style>

select {

text-indent: 29%;

text-align: center;

width: 350px;

height: 25px;

margin-bottom: 5px;

}

</style>

<input class="form-input" type="text" name="blood\_urea" placeholder="Enter your blood\_urea"><br>

<input class="form-input" type="text" name='blood glucose random' placeholder="Enter your blood glucose random"><br>

<select id="anemia" name="anemia">

<option value="">Select anemia or not</option>

<option value="1">YES</option>

<option value="0">NO</option>

</select><br>

<select id="coronary\_artery\_disease" name="coronary\_artery\_disease">

<option value="">Select coronary artery disease or not</option>

<option value="1">YES</option>

<option value="0">NO</option>

</select><br>

<select id="pus\_cell" name="pus\_cell">

<option value="">Select pus\_cell or not</option>

<option value="0">normal</option>

<option value="1">abnormal</option>

</select><br>

<select id="red\_blood\_cell" name="red\_blood\_cell">

<option value="">Select red\_blood\_cell level</option>

<option value="0">normal</option>

<option value="1">abnormal</option>

</select><br>

<select id="diabetesmellitus" name="diabetesmellitus">

<option value="">Select diabetesmellitus or not</option>

<option value="1">YES</option>

<option value="0">NO</option>

</select><br>

<select id="pedal\_edema" name="pedal\_edema">

<option value="">Select pedal\_edema or not</option>

<option value="1">YES</option>

<option value="0">NO</option>

</select><br>

<input type="submit" class="my-cta-button" value="Predict">

</form>

</div>

</body>

</html>

### RESULT PAGE:

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title>Chronic Kidney Disease</title>

<link rel="shortcut icon" href="{{ url\_for('static', filename='diabetes-favicon.ico') }}">

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='styles.css') }}">

<script src="https://kit.fontawesome.com/5f3f547070.js" crossorigin="anonymous"></script>

<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">

</head>

<body>

<!-- Website Title -->

<div class="container">

<h2 class='container-heading'><span class="heading\_font">Chronic Kidney Disease</span></h2>

<div class='description'>

<p>A Machine Learning Web App, Built with Flask</p>

</div>

</div>

<!-- Result -->

<div class="results">

{% if prediction\_text==1 %}

<h1>Prediction: <span class='danger'>Oops! You have Chronic Kidney Disease.</span></h1>

<img class="gif" src="{{ url\_for('static', filename='diabetes.webp') }}" alt="Diabetes Image">

{% elif prediction\_text==0 %}

<h1>Prediction: <span class='safe'>Great! You DON'T have Chronic Kidney Disease</span></h1>

<img class="gif1" src="{{ url\_for('static', filename='no-diabetes.webp') }}" alt="Not Diabetes Image">

{% endif %}

</div>

</body>

</html>

# CHAPTER-4 OUTPUTS

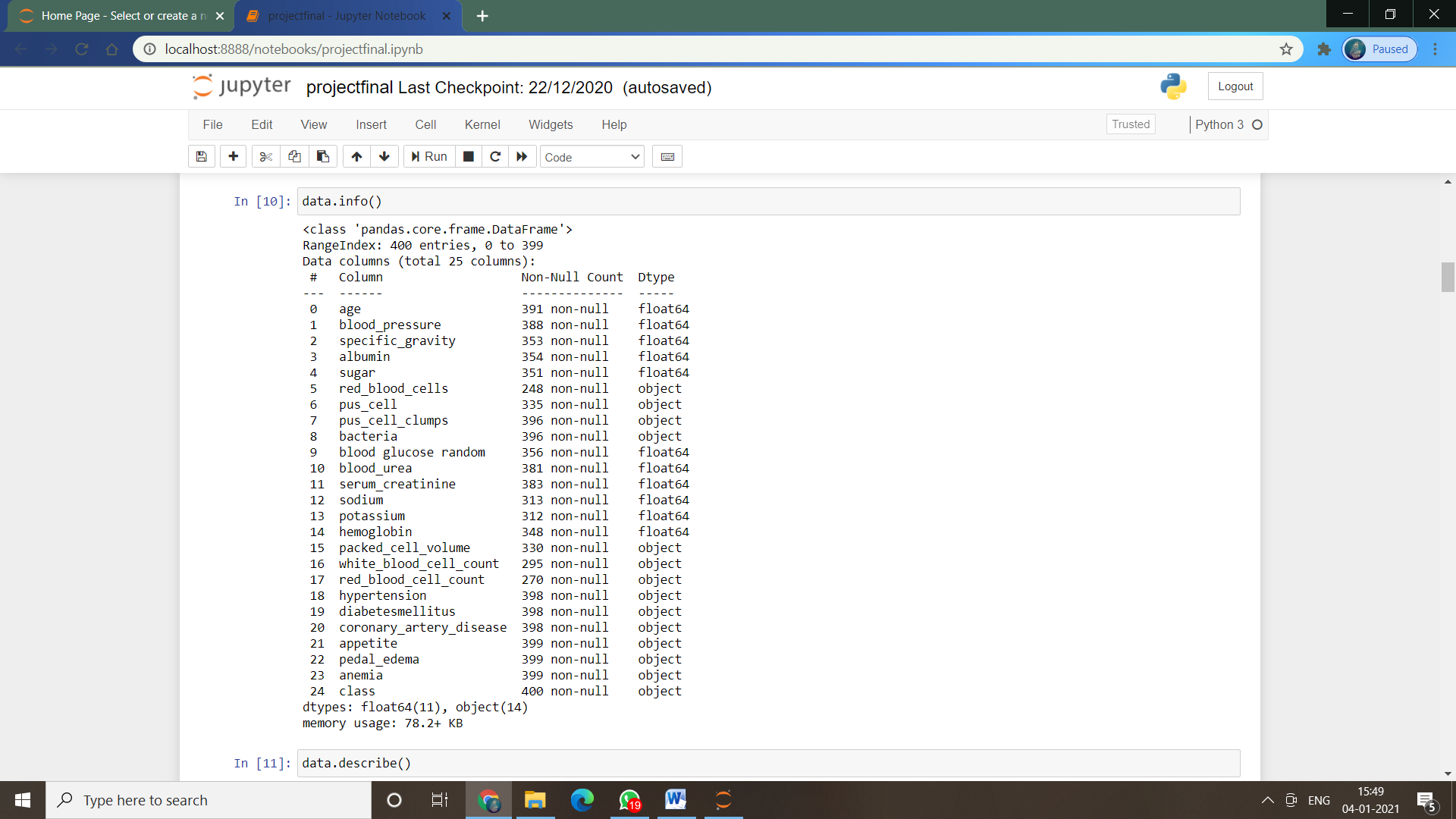


Fig.4.1 Information present in the dataset.

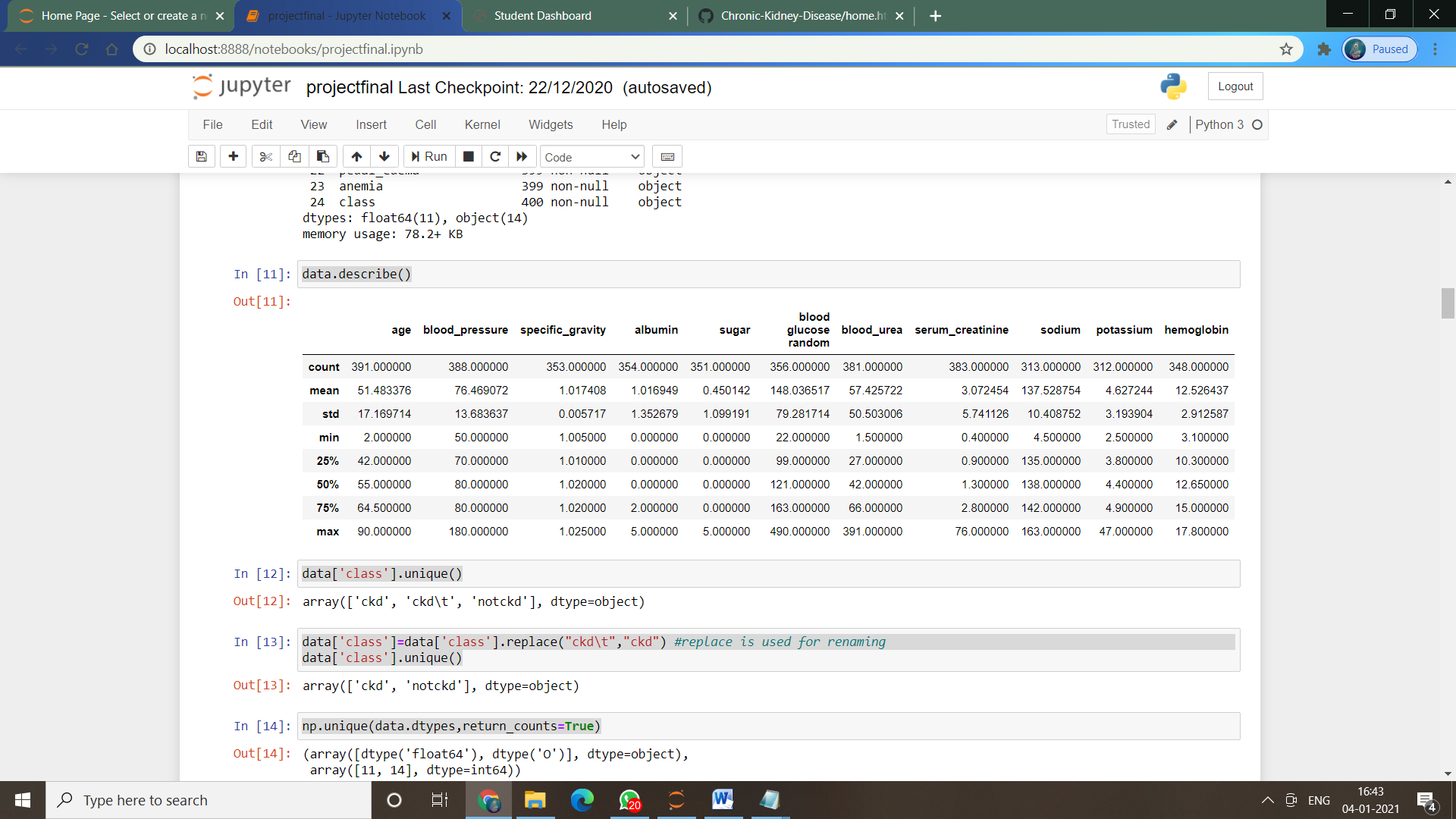


Fig.4.2 Describing the data.

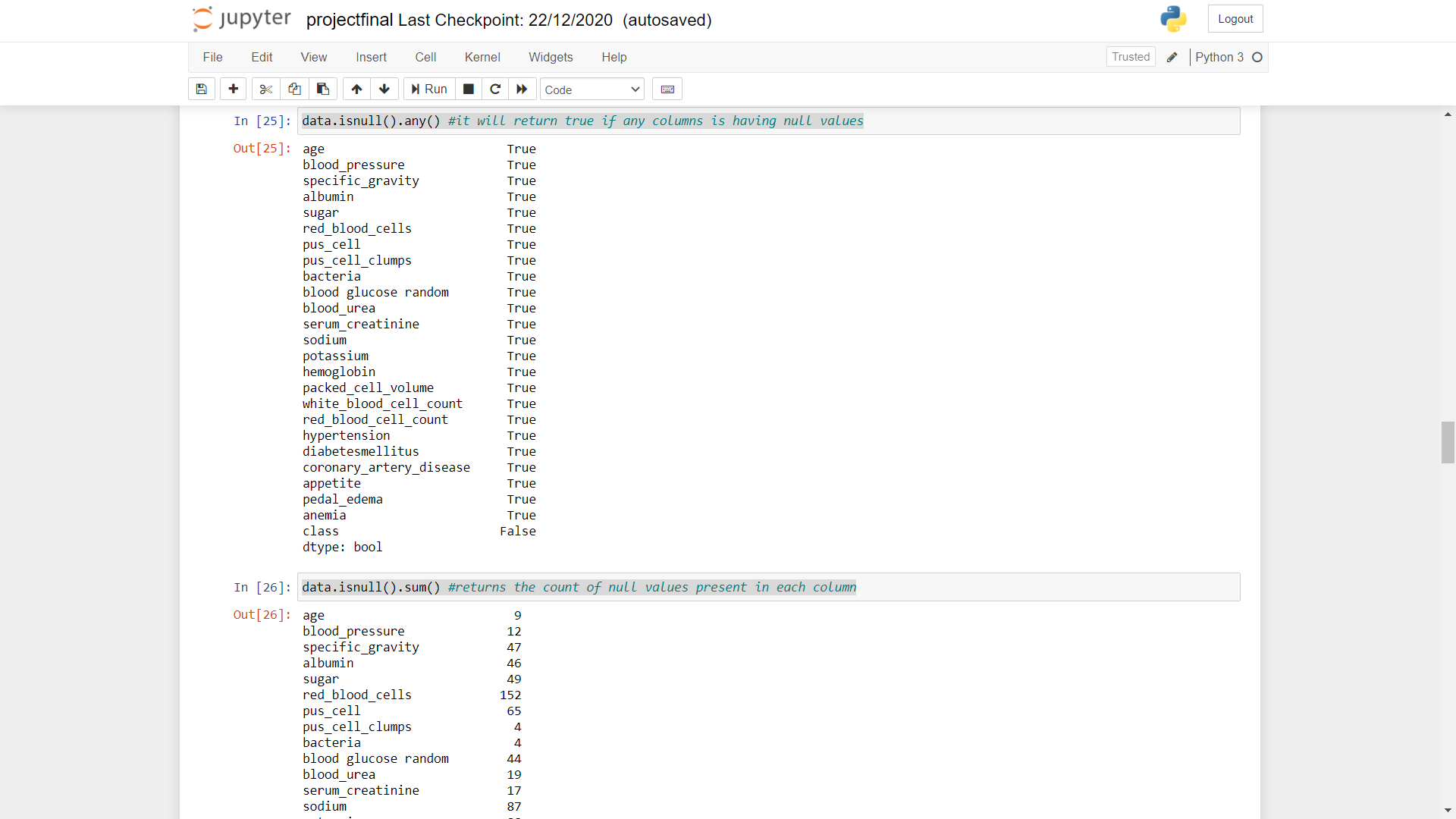


Fig.4.3 Null Values.

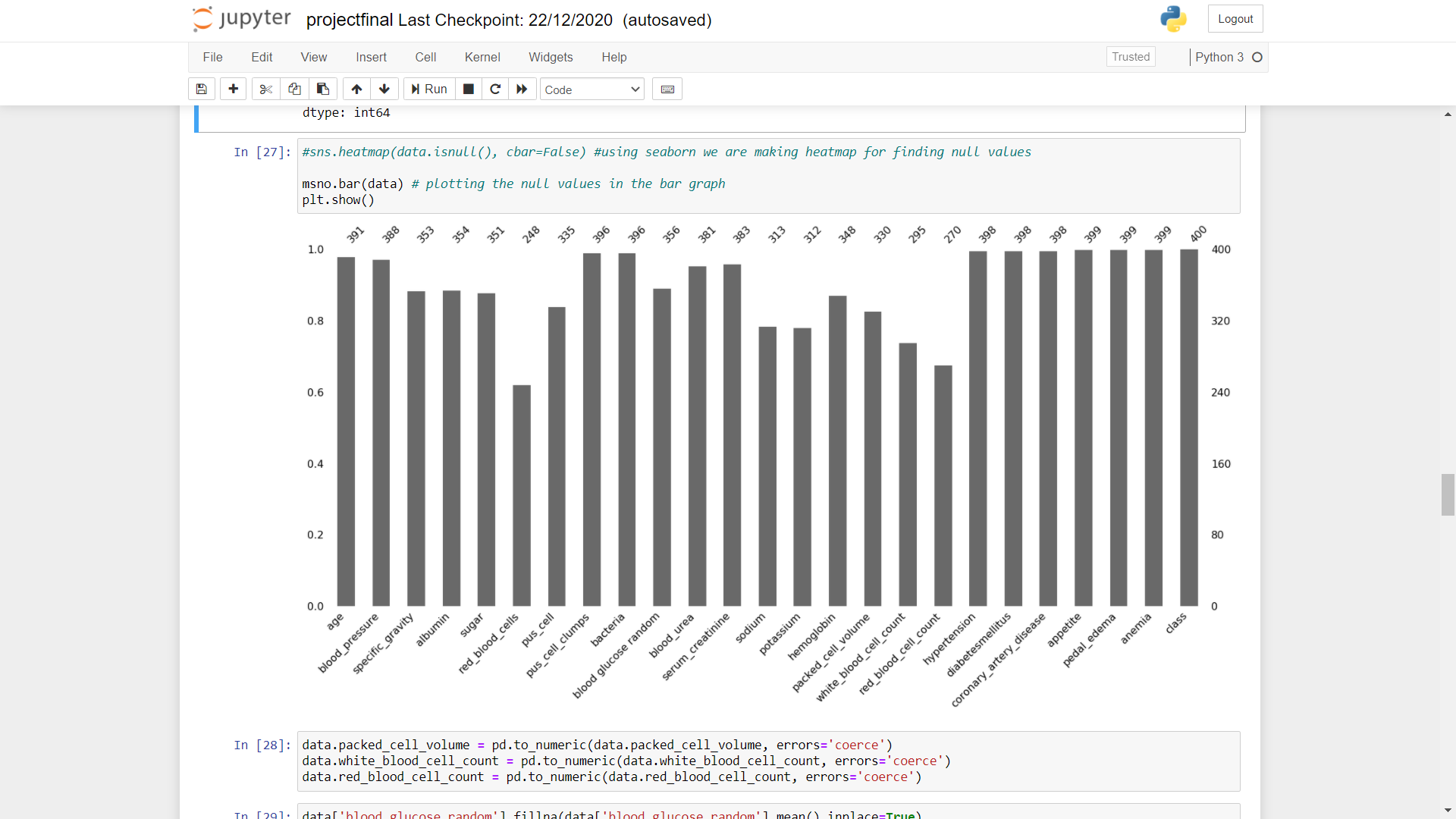


Fig.4.4 Heat map to find Null Values.

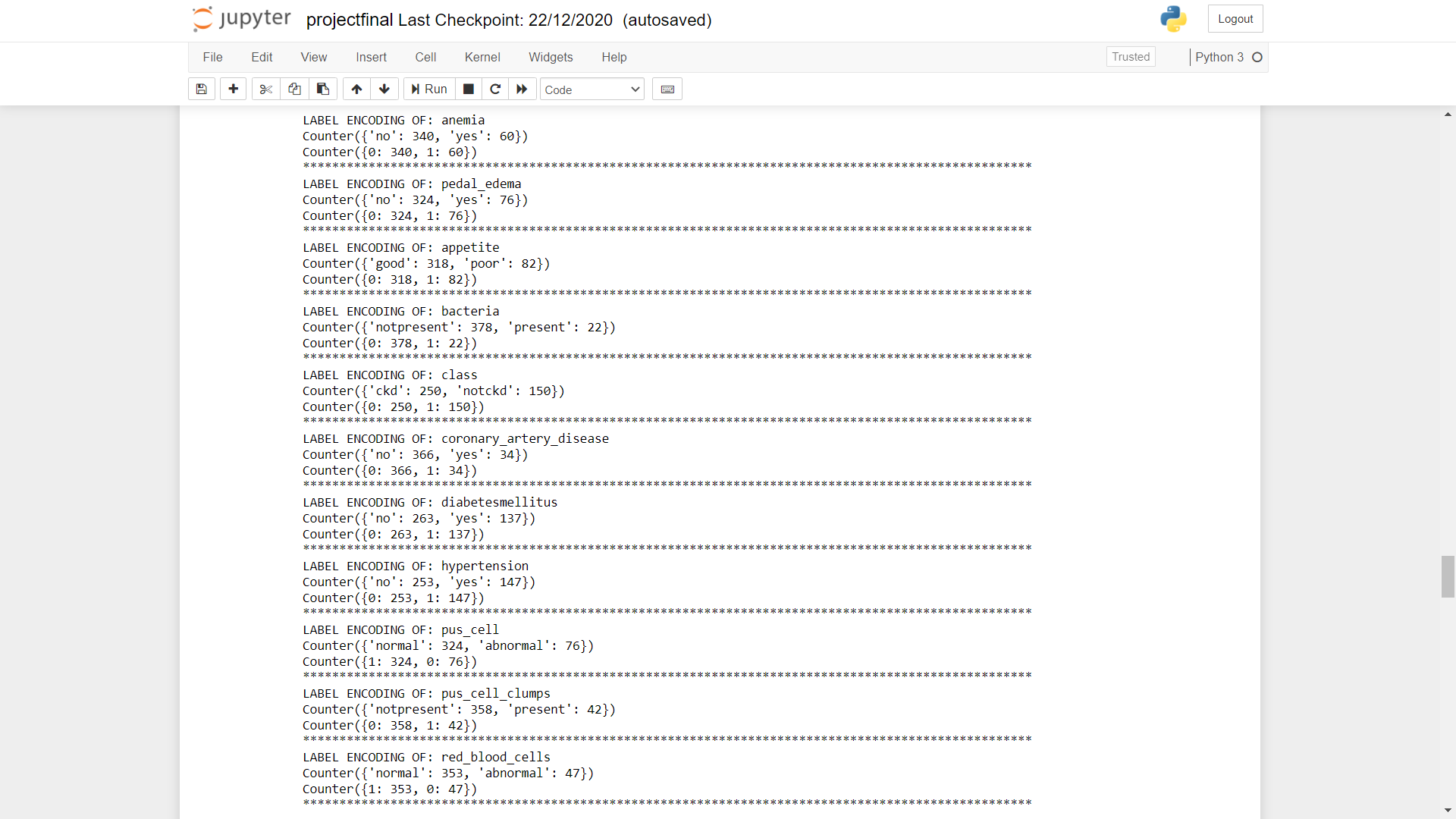


Fig.4.5 Encoding the data.

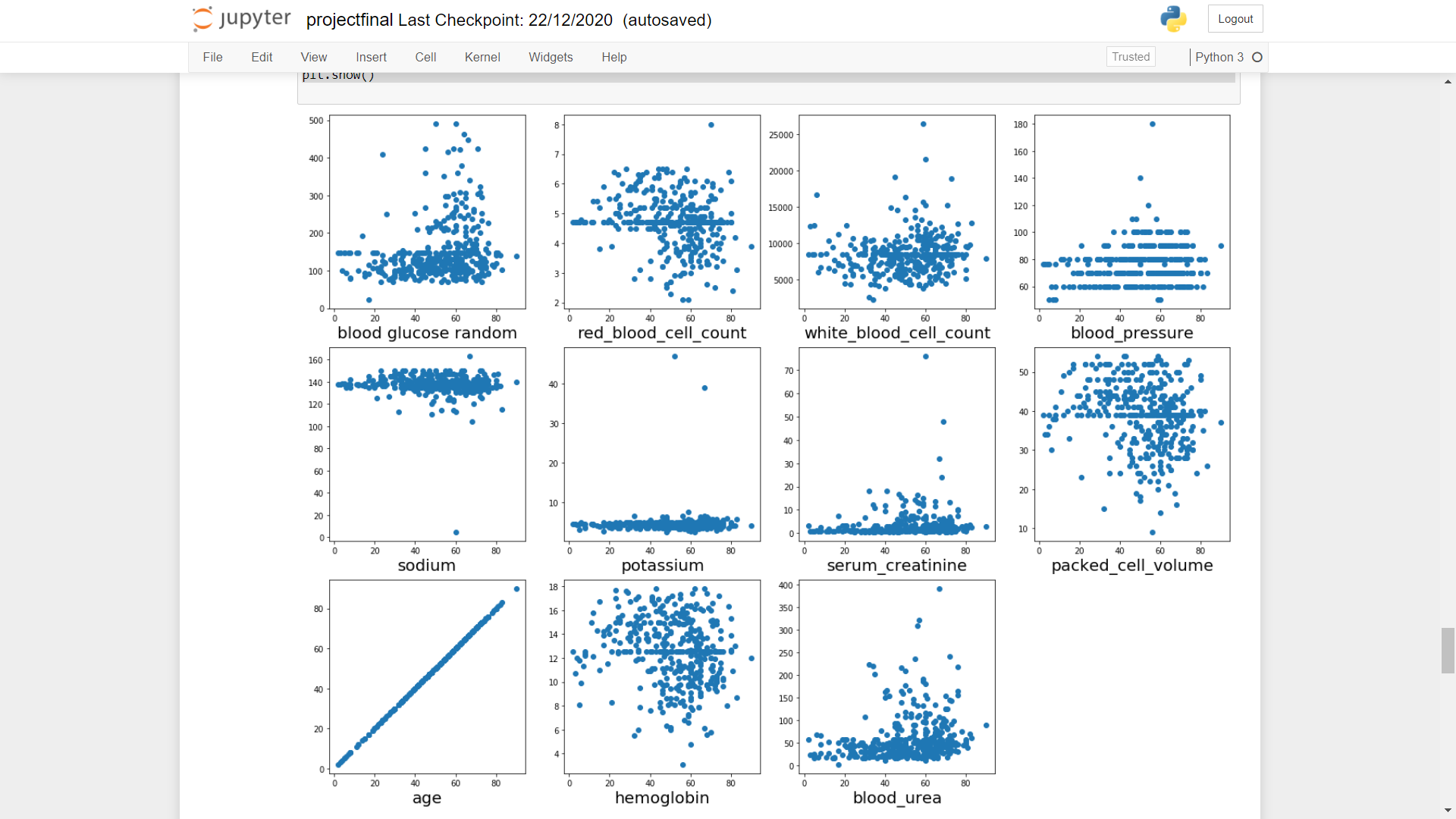


Fig.4.6 Scatter plots.

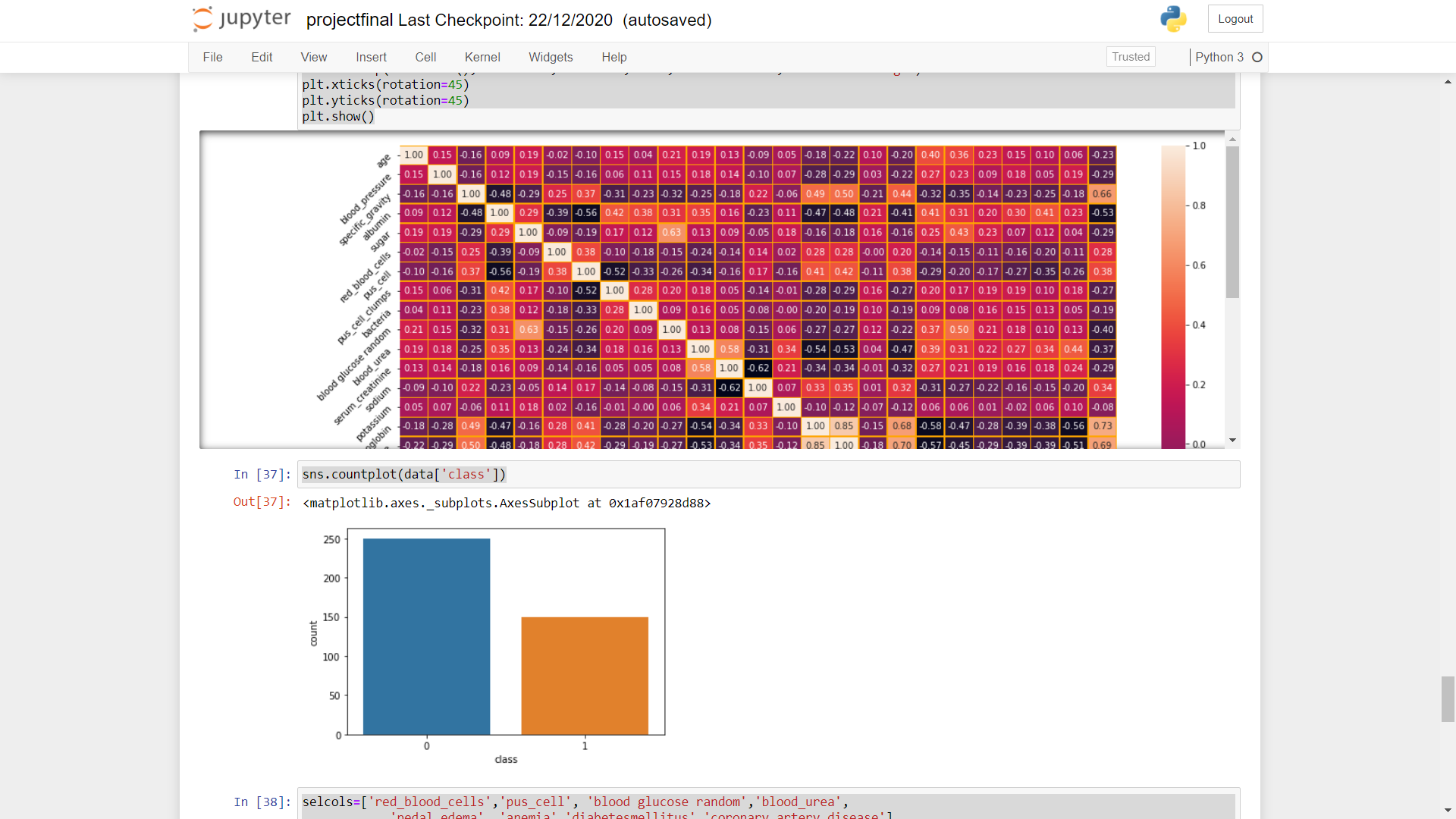


Fig.4.7 Correlation and bar graph of class Vs count.

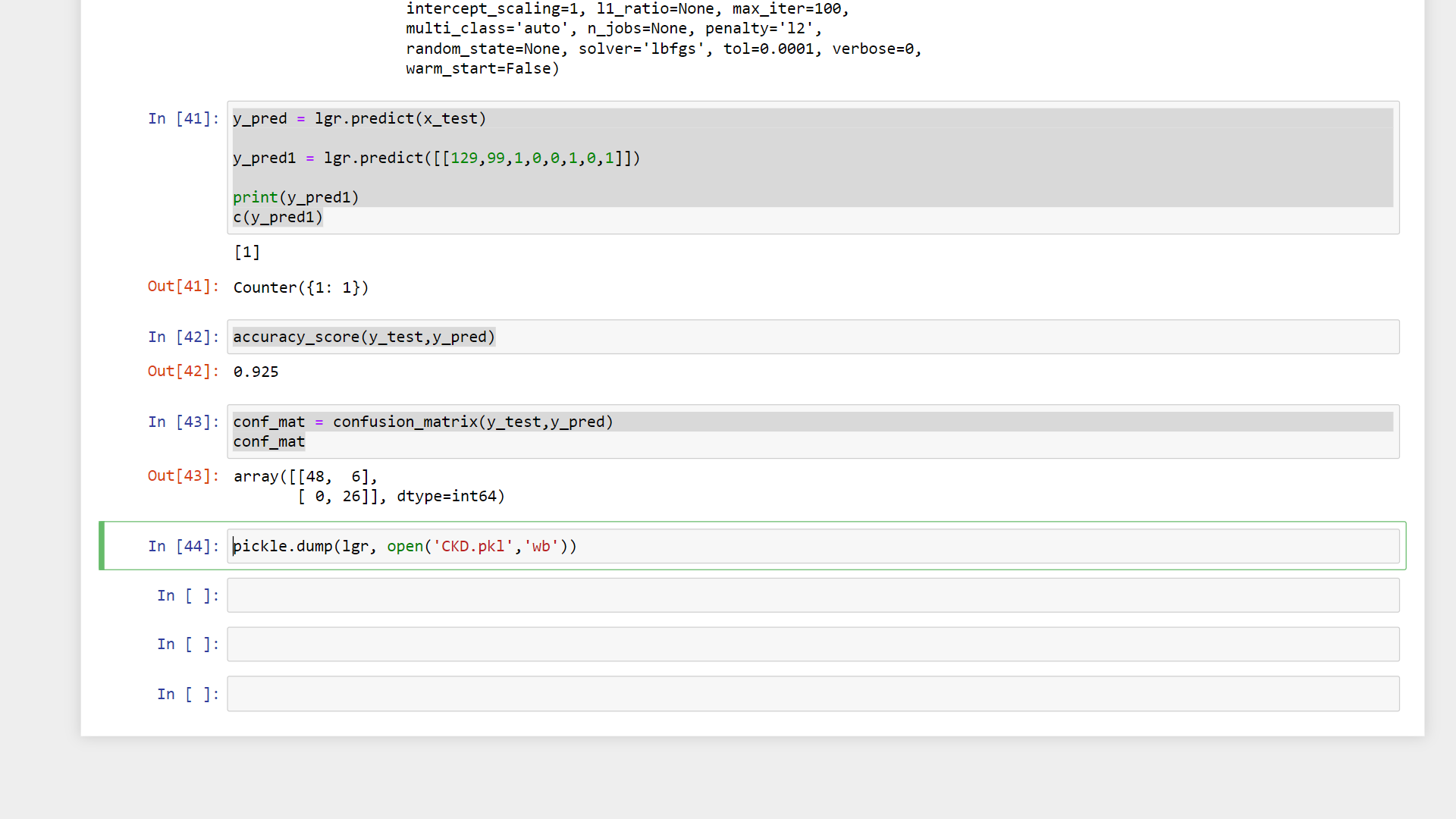


Fig.4.8 Accuracy Score

## 

Fig 4.9. Home page

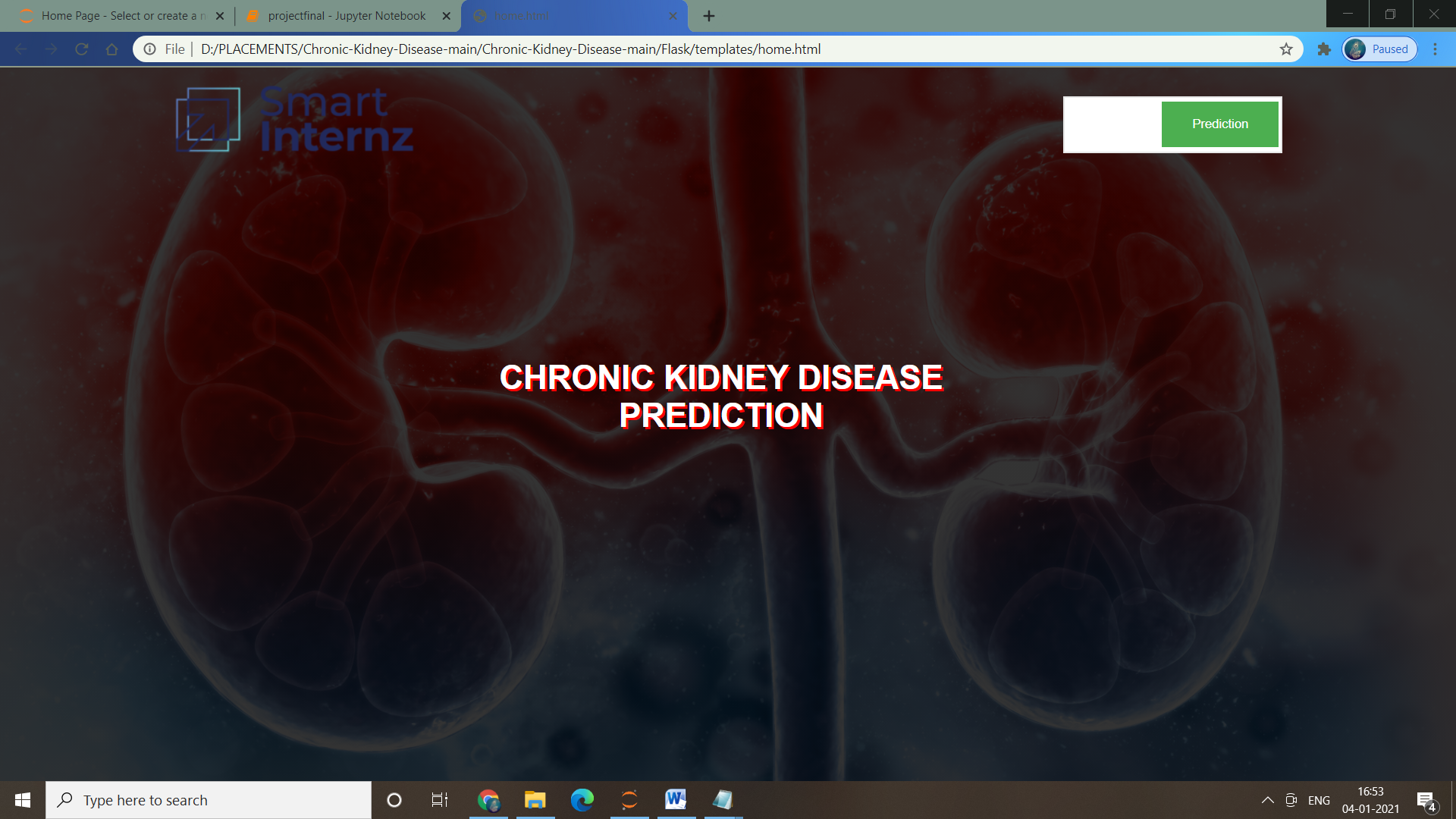


Fig.4.10. Index page.



Fig.4.11. Result page.

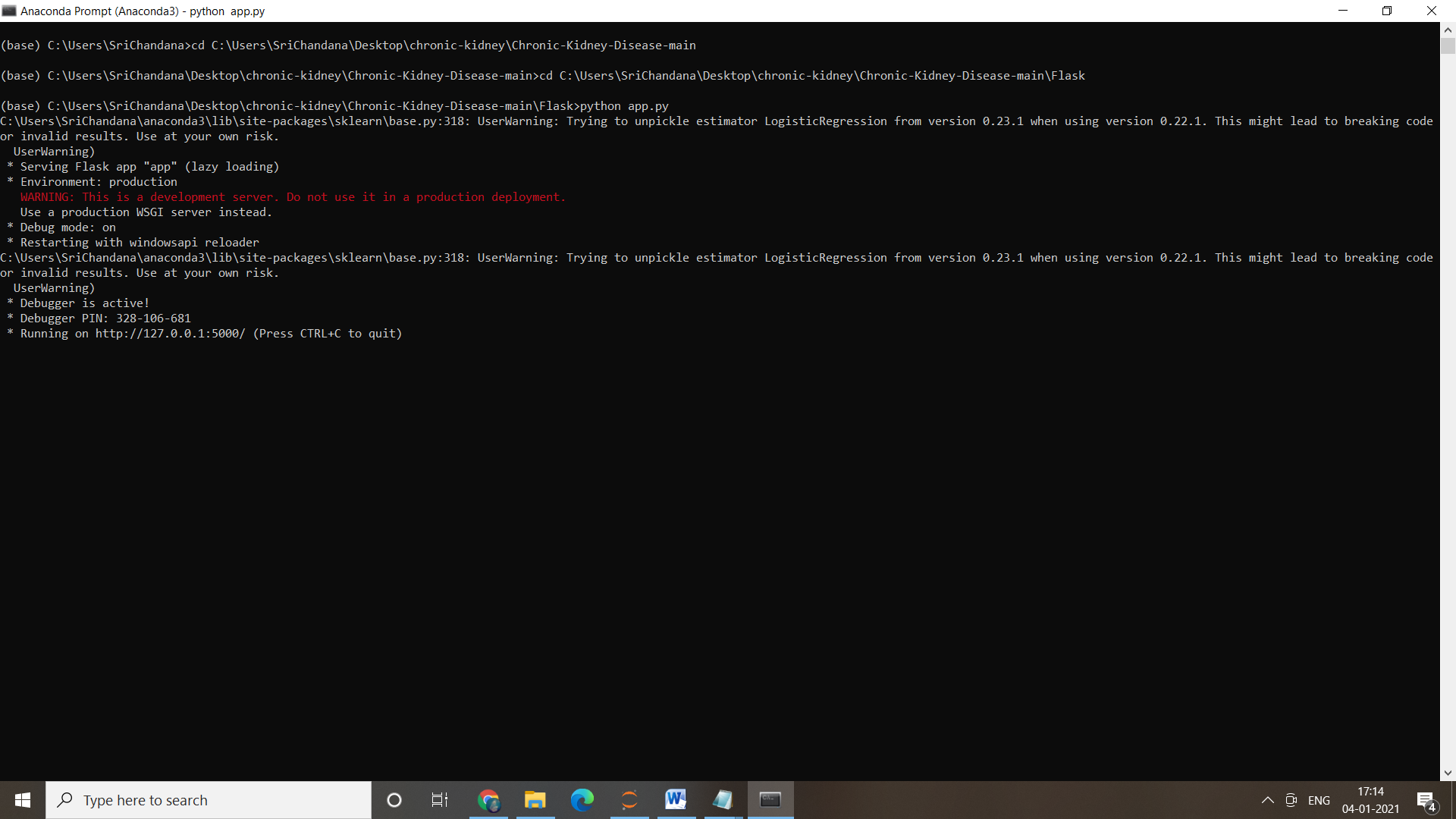


Fig 4.12.Running the App.

# CHAPTER-5 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.

# CHAPTER-6 REFERENCES

[1]. Gunarathne W.H.S.D,Perera K.D.M, Kahandawaarachchi K.A.D.C.P, “Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease (CKD)”,2017 IEEE 17th International Conference on Bioinformatics and Bioengineering.

[2]. S.Ramya, Dr. N.Radha, "Diagnosis of Chronic Kidney Disease Using Machine Learning Algorithms," Proc. International Journal of Innovative Research in Computer and Communication Engineering,Vol. 4, Issue 1, January 2016.

[3]. S.Dilli Arasu and Dr. R.Thirumalaiselvi, “Review of Chronic Kidney Disease based on Data Mining Techniques”,International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 23 (2017) pp. 13498-13505

[4]. L. Rubini, “Early stage of chronic kidney disease UCI machine learning repository,”2015. [Online].Available:http://archive.ics.uci.edu/ml/datasets/Chronic Kidney Disease.

[5]. S. A. Shinde and P. R. Rajeswari, “Intelligent health risk prediction systems using machine learning : a review,” IJET, vol. 7, no. 3, pp. 1019– 1023, 2018.

[6]. Himanshu Sharma,M A Rizvi,”Prediction of Heart Disease using Machine Learning Algorithms: A Survey”,International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169,Volume: 5 Issue: 8

[7]. Asif Salekin, John Stankovic, "Detection of Chronic Kidney Disease and Selecting Important Predictive Attributes," Proc. IEEE International Conference on Healthcare Informatics (ICHI), IEEE, Oct. 2016, doi:10.1109/ICHI.2016.36.

[8]. Pinar Yildirim, "Chronic Kidney Disease Prediction on Imbalanced Data by Multilayer Perceptron: Chronic Kidney Disease Prediction," Proc. 41st IEEE International Conference on Computer Software and Applications (COMPSAC), IEEE, Jul. 2017, doi: 10.1109/COMPSAC.2017.84

[9]. Sahil Sharma, Vinod Sharma, Atul Sharma, “Performance Based Evaluation of Various Machine Learning Classification Techniques for Chronic Kidney Disease Diagnosis,” July18, 2016