CHRONIC KIDNEY DISEASE DETECTION

AN INTERNSHIP PROJECT REPORT

By

Batch-5

Student Regd. No:21JG1A1251 **Student Name: S Divya Sriyani**

Under the esteemed guidance of

MS. M. DEEPTHI

ASSISTANT PROFESSOR

IT DEPARTMENT

Department of Information Technology

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation (**NBA**) for B.Tech. CSE, ECE & IT – Valid from 2019-22 and 2022-25]

[Accredited by National Assesment and Accreditation Council(**NAAC**)– Valid from 2022-27]

Kommadi , Madhurawada, Visakhapatnam–530048

2023–2024

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

DEPARTMENT OF INFORMATION TECHNOLOGY ENGINEERING

CERTIFICATE

This is to certify that the internship project report titled "Chronic kidney Detection Disease" is a bonafide work of following 3rd B.Tech. students in the Department of Information Technology, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2022-2023 Semester-II.

Student Regd. No:21JG1A1251 **Student Name: S Divya Sriyani**

Ms. M. Deepthi
Assistant Professor
(Internal Guide)

Dr. M. Bhanu Sridhar
Associate Professor
(Head of the department)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to **Ms. M. Deepthi,** Assistant Professor for encouragement all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We express our deep sense of gratitude and thanks to **Dr. M. Bhanu Sridhar, Assoc. Prof** and Head of the Department of Information Technology for her guidance and for expressing her valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We would like to take this opportunity to express our profound sense of gratitude to Vice Principal, **Dr. G. Sudheer** for allowing us to utilize the college resources thereby facilitating the successful completion of our project. We are also thankful to both teaching and non-teaching faculty of the Department of Information Technology for giving valuable suggestions for our project.

We would like to take the opportunity to express our profound sense of gratitude to the revered Principal, **Dr. R. K. Goswami** for all the help and support towards the successful completion of our project.

TABLE OF CONTENTS

ABSTRACT

Chronic kidney disease (CKD) is still a health concern despite advances in surgical care and treatment. CKD’s growth in recent years has gained much interest from researchers around the world in developing high-performance methods for diagnosis, treatment and preventive therapy. Improved performance can be accomplished by learning the features that are in the concern of the problem. In addition to the clinical examination, analysis of the medical data for the patients can help the health care partners to predict the disease in early stage. Although there are many tries to build intelligent systems to predict the CKD by analysis the health data, the performance of these systems still need enhancement. This research can help us to better understand global disparities in kidney disease, as well as what we can do to address them and coordinate our efforts to achieve global kidney health equity. This study provides an excellent feature-based prediction model for detecting kidney disease. Various machine learning algorithms, including, Random forest (RF), k-nearest neighbours algorithm (KNN), artificial neural networks (ANN), support vector machines (SVM), Gaussian Naive Bayes (GNB), Logistic regression. The studies found that a logistic regression-based prediction model with optimal features chosen using the Chi-Square technique had the highest accuracy of 98.75 percent. White Blood Cell Count (Wbcc), Blood Glucose Random (bgr), Blood Urea (Bu), Serum Creatinine (Sc), Packed Cell Volume (Pcv), Albumin (Al), Hemoglobin (Hemo), Age, Sugar (Su), Hypertension (Htn), Diabetes Mellitus (Dm), and Blood Pressure (Bp) are examples of these traits.

1.Introduction

An early diagnosis of such CKD is crucial and can save a patient’s life. Medical experts utilized several fundamental approaches to gather precise insights about renal disease identification, such as medical examinations and lab tests (i.e., blood tests and urine tests). The blood test determines the glomerular filtration rate (GFR), which could be used to indicate kidney function. The urine test is used to determine albumin level, indicating whether the kidney is working correctly. With promising data sources that could help in medical diagnosis, developing robust and generalized diagnosis models that could assist medical experts and give accurate and timely decisions is crucial. Recently, machine learning (ML) has contributed to developing effective models in the medical diagnosis domain, which could make accurate and timely decisions. Deep learning (DL) is a subsection of ML that seeks to find underlying links within a dataset through a set of operations that occurred while training. DL is a multilayer DL model that could theoretically handle nonlinear data; it significantly affects medical applications. For example, Feng long et al. utilized DL to develop a predictive model that predicts CKD based on several medical examination data. The same statement applies to Unless DL progresses in various applications, it has several limitations due to the heterogeneity of the medical data, which exacerbates the generalization and robustness of the developed model, This challenge could be overcome by using diverse and various DL models.

1.1 Problem Statement:

Kidney disease has become a typical disease with severe problems . Kidney disease is described as a heterogeneous cluster of disorders affecting the structure and performance of the urinary organ There are four stages of kidney diseases: 1. NO Kidney Disease (NKD) 2. Chronic Kidney Disease (CKD) 3. Acute Kidney Injury (AKI) 4. End Stage Kidney Disease (ESKD) Around 10% of the inhabitancy in the world suffers from chronic kidney ailment (CKD), leading to millions of deaths every year Chronic Kidney Disease is a significant health issue worldwide, affecting millions of people. Early detection and intervention are critical to manage CKD effectively and improve patient outcomes. Machine learning can play a pivotal role in predicting CKD based on patient data, thereby allowing healthcare professionals to make timely decisions. Various machine learning algorithms, including, Random forest (RF+

), k-nearest neighbours algorithm (KNN), artificial neural networks (ANN), support vector machines (SVM), Gaussian Naive Bayes (GNB), Logistic regression.

SVC() and LogisticRegres() both achieved an accuracy of 0.9875. This is very high accuracy, and it suggests that both models are able to learn the underlying patterns in the data and make accurate predictions. Data set name “chronic_kidney_disease.csv

1.2 Objectives:

The main factors specified as a reason for the disease are Obesity and hypertension: Many medical conditions can cause KD. Family history: If anyone in your family has kidney disease, dialysis, or kidney transplantation, you may be more likely to develop kidney disease than someone without this family history. Medicines: Some medicines can cause or exacerbate kidney disease, such as over-the-counter pain medicines. Age and race: older people and certain racial groups may have a higher chance of developing renal disease.

The diagnosis of kidney disease in early stage saves the patient from serious complications. To predict the kidney diseases, the factors that cause it must be studied carefully. All these factors can be translated into data to predict kidney disease and suggest a medical protocol to improve the patient health state. The problem of using large volumes of data efficiently is becoming a major issue for the healthcare industry . Nowadays, in health care domain, data mining plays an indispensable role in disclosing anonymous and valuable knowledge in health data. Health care partners can ameliorate quality of service by identifying latent, potentially valuable patterns that medical diagnosis demands by applying data mining techniques to health care domain . Classification is one of healthcare organization’s most utilized forms for data mining. The classification method determines the target category for each set of data.

Classification is a valuable data mining process that suggests classification system (called classifier) to new cases. Recently, in the domain of kidney disease, Deep Learning (DL) methods are used to automate the extraction and interpretation of the functions, hence models based on these techniques achieve high performance . Deep learning is a machine learning focuses on learning to describe and construct multiple levels. Deep Neural Network (DNN) is an input-output neural network with multiple layers of nodes. It Identifies and processes series of layers between input and output in several stages. Deep learning requires computational models consisting of multiple layers of storage to acquire data structures of different abstraction scales. Deep learning approaches aim to learn features in hierarchical structures where features at higher hierarchy levels are created by lower-level features composition .

2.Related work

Literature Survey:

2.1 Existing System:

Title: Chronic kidney disease detection using statistical methods

Abstract

Chronic kidney disease (CKD) is a progressive condition that can lead to kidney failure. Early detection and intervention are essential for preventing or delaying the progression of CKD. However, CKD is often asymptomatic in its early stages, making it difficult to diagnose.

Statistical methods can be used to identify patterns in patient data that are associated with CKD. In this paper, we propose a statistical method for CKD detection. The proposed method uses a combination of univariate and multivariate analysis to identify risk factors for CKD.

Methods

The proposed method consists of the following steps:

1. Univariate analysis: Univariate analysis is used to identify individual features that are associated with CKD. This is done by calculating the mean, standard deviation, and p-value for each feature in patients with and without CKD.

2. Multivariate analysis:Multivariate analysis is used to identify combinations of features that are associated with CKD. This is

2.2 Proposed System:

Early detection and diagnosis: Machine learning algorithms can be used to identify individuals at risk of developing CKD, even before they show any symptoms. This can lead to earlier diagnosis and treatment, which can help to slow the progression of the disease.

Improved accuracy: Machine learning algorithms can be used to improve the accuracy of CKD diagnosis. This is because they can consider a wide range of factors, such as patient demographics, medical history, and laboratory test results.

Personalized risk assessment: Machine learning algorithms can be used to generate personalized risk assessments for CKD. This can help patients to understand their individual risk of developing the disease and to make informed decisions about their care.

Identification of new risk factors: Machine learning algorithms can be used to identify new risk factors for CKD. This can help to improve our understanding of the disease and to develop new strategies for prevention.

Improved prognosis: Machine learning algorithms can be used to predict the prognosis of CKD patients. This can help patients and their families to make informed decisions about their care.

Reduced costs: Machine learning algorithms can be used to reduce the costs associated with CKD diagnosis and management. For example, machine learning algorithms can be used to identify patients who are at high risk of complications, so that they can be monitored more closely. This can help to prevent the need for costly hospital stays.

Improved patient outcomes: By using machine learning algorithms to improve the diagnosis, prognosis, and management of CKD, we can improve patient outcomes. For example, machine learning algorithms can be used to identify patients who are likely to respond to specific treatments. This can help to ensure that patients receive the most effective treatment possible.

In addition to the advantages listed above, machine learning algorithms are Random forest (RF), k-nearest neighbors algorithm (KNN), artificial neural networks (ANN), support vector machines (SVM), Gaussian Naive Bayes (GNB), Logistic regression be used to:

Identify patients who are likely to develop complications from CKD.

Monitor the progression of CKD.

Develop new treatments for CKD.

Overall, machine learning algorithms have the potential to significantly improve the management of CKD. By using these algorithms, we can improve the diagnosis, prognosis, treatment, and outcomes of the disease.

3. System analysis

3.1 Software Requirement Specification:

Feasibility study:

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

Economical Feasibility

Technical Feasibility

Social Feasibility

Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 System Requirements:

3.2.1 Hardware Requirements

- System : MINIMUM i3.
- Hard Disk : 40 GB.
- Ram : 4 GB.

3.2.2 Software Requirements

- **Operating System:** Windows 8
- **Coding Language**: Python 3.7

Python:

Python is a high level general purpose open source programming language. It is both object oriented and procedural. Python is an extremely powerful language. This language is very easy to learn and is a good choice for most of the professional programmers.

Python is invented by **Guido Van Rossum** at CWI in Netherland in 1989. It is binding of **C, C++, and Java**. It also provides a library for GUI.

Python Features and Characteristics:

❖ Python is a high level, open source, general purpose programming language.
❖ It is object oriented, procedural and functional.
❖ It has library to support GUI.
❖ It is extremely powerful and easy to learn.
❖ It is open source, so free to available for everyone.
❖ It supports on Windows, Linux and Mac OS.
❖ As code is directly compiled with byte code, python is suitable for use in scripting languages.
❖ Python enables us to write clear, logical applications for small and large tasks.
❖ It has high level built-in data types: string, lists, dictionaries etc.
❖ Python has a set of useful Libraries and Packages that minimize the use of code in our day to day life, like- Django, Tkinter, OpenCV, NextworkX, lxml.

- **PYPY –** Python with JIT compiler and stackless mode.
- **IronPython –** Python for Net and CLR.

Jupyter Notebook

A Jupyter Notebook is a versatile tool for working with Chronic Kidney Disease (CKD) data in Python. It combines code, visualizations, and explanatory text in an interactive environment. Here's an explanation of how you can use Jupyter Notebook for CKD analysis and some of its key features:

4. System design

4.1 Introduction:

1. Data Collection:

Collect relevant data on patients, including demographic information, medical history, lab test results, and other clinical parameters. You can obtain data from publicly available datasets, electronic health records, or through collaborations with healthcare institutions.

2. Data Preprocessing:

Clean and preprocess the data to handle missing values, outliers, and format inconsistencies. This step may include data normalization, feature engineering, and encoding categorical variables.

3. Feature Selection:

Select the most relevant features for prediction to reduce dimensionality and improve model performance. Feature selection methods such as Recursive Feature Elimination (RFE) or feature importance from tree-based models can be used.

4. Model Selection:

Choose appropriate machine learning models for predicting chronic kidney disease. Common models include logistic regression, decision trees, random forests, support vector machines, and gradient boosting.

5. Model Training:

Split the dataset into training and testing subsets. Train the selected models using the training data and optimize their hyperparameters to achieve the best performance. Cross-validation can help evaluate the model's generalization ability.

6. Model Evaluation:

Evaluate the model's performance using appropriate evaluation metrics. For binary classification problems like predicting chronic kidney disease, you can use metrics such as accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC-ROC).

7. Model Interpretability:

Ensure that the chosen model is interpretable to healthcare professionals. You can use techniques like feature importance analysis, SHAP values, or LIME (Local Interpretable Model-Agnostic Explanations) to interpret the model's predictions.

8. Deployment:

Deploy the trained model as a standalone script or a command-line tool. This allows users to input patient data and receive predictions without the need for a web application. Alternatively, you can integrate it into existing healthcare systems**.**

9. Data Security and Compliance:

Ensure that the system complies with data privacy and security regulations (e.g., HIPAA) to protect sensitive patient information.

10. Monitoring and Maintenance:

Regularly monitor the model's performance in a real-world setting. Update the model as needed to account for changes in the patient population, new research, or improvements in model performance.

11. Documentation:

Document the entire system, including data sources, preprocessing steps, model selection, and evaluation results. This documentation will be invaluable for transparency and troubleshooting.

12. Knowledge Transfer:

Train healthcare professionals on how to use the system effectively and interpret its results.

13. Continuous Improvement:

Continuously refine the model and the system by incorporating new data and research findings to enhance its predictive accuracy and utility.

5. Methodology

5.1 Architecture Diagram

System Architecture:

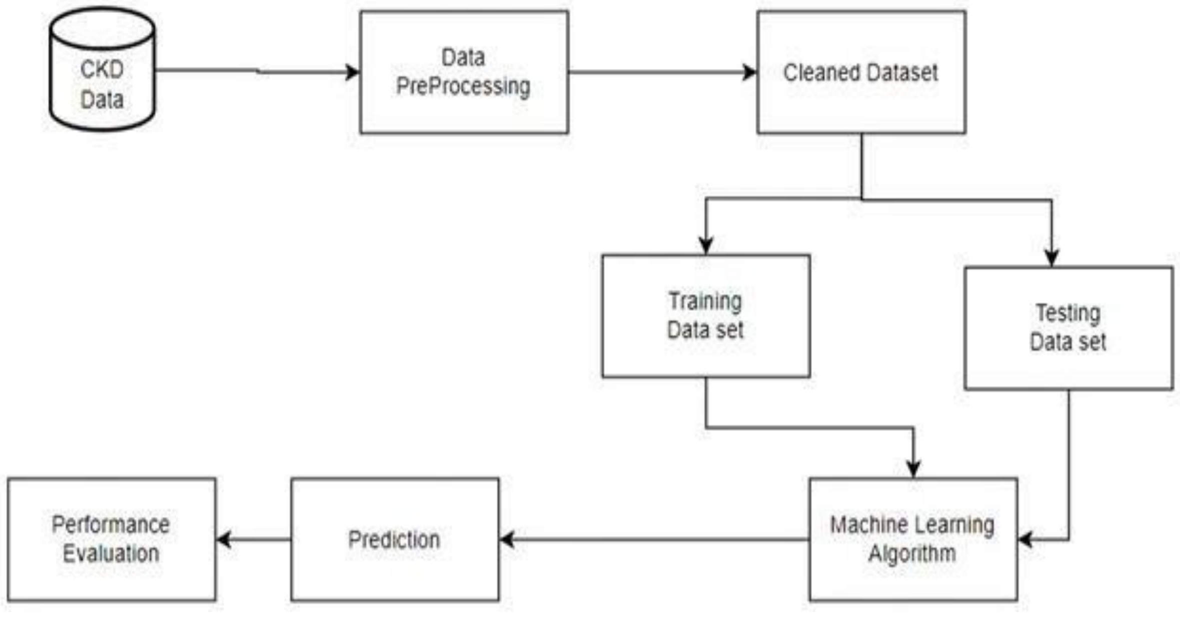
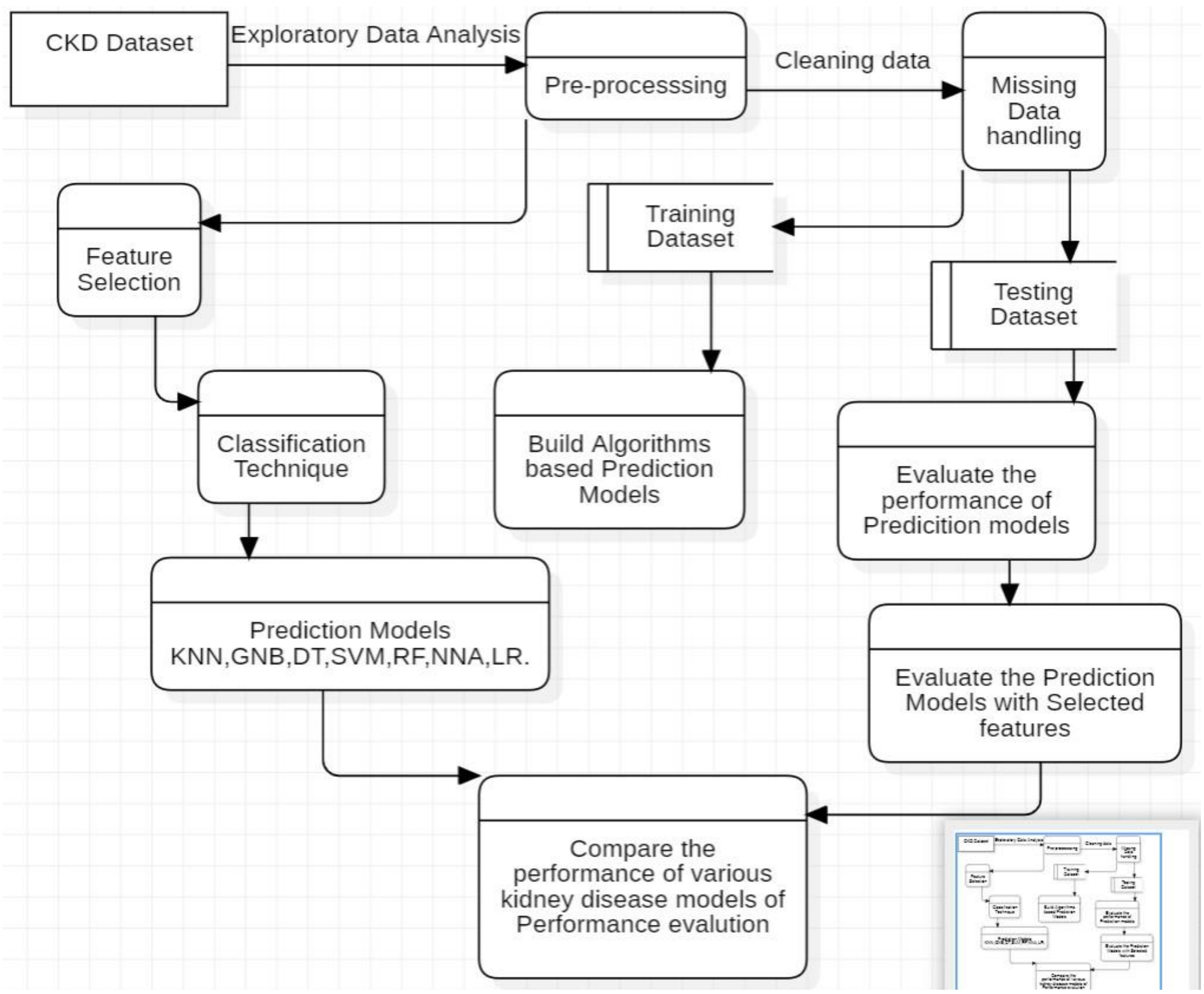


Fig 5.1

Data Flow Diagram:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing informationflow and functional detail.



CKD Dataset
Exploratory Data Analysis
Pre-processsing
Cleaning data
Missing Data handling
Feature Selection
Training Dataset
Testing Dataset
Classification Technique
Build Algorithms based Prediction Models
Evaluate the performance of Predicition models
Prediction Models KNN,GNB,DT,SVM,RF,NNA,LR.
Evaluate the Prediction Models with Selected features
Compare the performance of various kidney disease models of Performance evalution

Fig:5.2

5.2 Modules , Descriptions

To predict chronic kidney disease, this study employs Decision Tree Classifiers, Random Forest Classifiers, Support Vector Machines, and Artificial Neural Networks. Among these algorithms, we attempt to construct our prediction model, and we choose the best performance by assessing their accuracy.

1. Support Vector Machine (SVM)

This is the most well-known and useful supervised machine-learning method, which works on classification and regression issues but is most used for classification. To segregate labelled data, SVM employed a kernel function. One of the benefits of employing kernels in SVM is that SVM applies kernel dentitions to non-vector inputs, and kernels may be built using a variety of data types. The SVM algorithm divides data into two data points when the hyperplane sits between two branches. SVM generates a distinct hyper plane in the training data's signifier space, and compounds are categorized based on which side of the hyper plane they are found on. SVM has been used in a variety of applications, including bioinformatics and handwriting recognition. Other uses for SVM include medical diagnosis, weather prediction, financial market analysis, and image processing. SVM, like all other machine learning approaches, is a computer algorithm that assigns labels to objects based on experience and examples.

2. The Decision Tree

The decision tree algorithm is a supervised machine learning technique that can handle both classification and regression issues, however it is most employed to tackle classification problems. The Decision Tree approach solves the classification issue by turning the dataset into a tree representation through feature value sorting. Every node in a decision tree represents a feature of an instance to be categorized, and every leaf node represents a class label to which the instance belongs.

3. Random Forest

The random forest technique is a simple and adaptable supervised machine learning approach that uses diverse collections of decision trees to solve classification and regression problems. Random forest works by constructing the forest, which is an ensemble of decision trees that are typically trained using the bagging approach. Bagging techniques are a method of combining learning models to improve overall outcomes. This model consists of numerous decision trees and outputs the class target that is the target with the highest selection outcomes from each tree.

4. Neural Network Artificial

Artificial Neural Networks, or ANN, are essentially computational models. These computations Models are composed of a sophisticated network of fundamental components or nodes known as neurons. The nodes are connected to one another. Each node-to-node link has a weight associated with it. A neural network's fundamental structure consists of three layers. The first layer is referred to as the input layer, the intermediate layer as the concealed layer, and the last layer as the output layer. All three layers of a neural network will almost certainly include one or more nodes. A rudimentary neural network will only have one hidden layer. A sophisticated neural network, on the other hand, may include numerous hidden layers. The network's inputs are routed through the basic input layers. The calculation takes place in the hidden layers, and the resulting output is sent through the output layers. There are several types of neural networks accessible.

5. Logistic Regression:

Logistic regression is a simple but effective machine learning model that can be used for both classification and regression tasks. It is often used as a baseline model to compare the performance of other machine learning models.It models the probability that a given input belongs to a particular class. It's a linear model with a logistic (sigmoid) function applied to the output.

6. Gaussian Naive Bayes (GNB):

Gaussian Naive Bayes (GNB) is a simple but effective classification algorithm that is based on Bayes' theorem. GNB assumes that the features in the data are independent, which is often not the case with CKD data. However, GNB can still be effective for CKD prediction if the features are carefully selected. Naive Bayes is a probabilistic algorithm often used in text classification, but it can also be applied to other classification tasks.

7.k-nearest neighbors (KNN):

k-nearest neighbors (KNN) is a simple classification algorithm that works by finding the k most similar data points to the new data point and then predicting the class of the new data point based on the classes of the k most similar data points. KNN can be effective for CKD prediction, but it can be computationally expensive to train and predict on large datasets. KNN is a simple instance-based algorithm that classifies a patient based on the majority class among its k-nearest neighbors in the feature space.

6.Implementation and code with comments

Source code

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import math

import warnings

warnings.filterwarnings("ignore")
```

#Pre-processing

#A bit of exploration

```
feature_names=['Age (yrs)','Blood Pressure (mm/Hg)','Specific Gravity','Albumin','Sugar','Red Blood Cells','Pus Cells','Pus Cell Clumps','Bacteria','Blood Glucose Random (mgs/dL)','Blood Urea (mgs/dL)','Serum Creatinine (mgs/dL)','Sodium (mEq/L)','Potassium (mEq/L)','Hemoglobin (gms)','Packed Cell Volume','White Blood Cells (cells/cmm)','Red Blood Cells (millions/cmm)','Hypertension','Diabetes Mellitus','Coronary Artery Disease','Appetite','Pedal Edema','Anemia','Chronic Kidney Disease']

data=pd.read_csv("chronic_kidney_disease.csv", names = feature_names)

data.head()
```

#We can see there are missing values.

If their number is considerable, then we would have to be careful about which imputation technique to use.

```
data.info()

data.describe()
```

###You can see that some features have some serious outliers.

###Data Cleaning

###Let's deal with typos first.Correcting some typos in the dataset

```
for i in range(data.shape[0]):

    if data.iloc[i,24]=='ckd\t':

        data.iloc[i,24]='ckd'

    if data.iloc[i,19] in [' yes','\tyes']:

        data.iloc[i,19]='yes'

    if data.iloc[i,19]=='\tno':
        data.iloc[i,19]='no'
    if data.iloc[i,20]=='\tno':
        data.iloc[i,20]='no'
    if data.iloc[i,15]=='\t?':
        data.iloc[i,15]=np.nan
    if data.iloc[i,15]=='\t43':
        data.iloc[i,15]='43'
    if data.iloc[i,16]=='\t?':
        data.iloc[i,16]=np.nan
    if data.iloc[i,16]=='\t6200':
        data.iloc[i,16]= '6200'
    if data.iloc[i,16]=='\t8400':
        data.iloc[i,16]= '6200'
    if data.iloc[i,17]=='\t?':
        data.iloc[i,17]=np.nan
    if data.iloc[i,24]=='ckd':
        data.iloc[i,24]='yes'
    if data.iloc[i,24]=='notckd':
        data.iloc[i,24]='no'
```

#One-hot encoding some categorical features, some categorical features must not be encoded since they are ordinal

```
data.replace({ 'normal' : 1, 'abnormal' : 0}, inplace = True)
data.replace({ 'present' : 1, 'notpresent' : 0}, inplace = True)
data.replace({ 'yes' : 1, '\tyes':1, ' yes':1, '\tno':0, 'no' : 0}, inplace = True)
data.replace({ 'good' : 1, 'poor' : 0}, inplace = True)
data.replace({ 'ckd' : 1, 'ckd\t':1, 'notckd' : 0}, inplace = True)
```

###Replacing missing values with NaN

```
data.replace('\t?',np.nan, inplace = True)
data.replace('?',np.nan, inplace = True)
data.head()
```

###Let's deal with mistyped features now.

```
data.dtypes
```

###Some numerical features are mistyped as strings.

```
for col in data.columns:
        data[col]=data[col].astype('float')
data.info()
```

###Now that we've dealt with that, let's separate categorical and numerical features, as they won't be dealt with the same way.

###Creating two lists of numerical and categorical features

```
numeric = ['Age (yrs)','Specific Gravity','Albumin','Sugar', 'Blood Pressure (mm/Hg)', 'Blood Glucose Random (mgs/dL)', 'Blood Urea (mgs/dL)', 'Serum Creatinine (mgs/dL)', 'Sodium (mEq/L)', 'Potassium (mEq/L)', 'Hemoglobin (gms)', 'Packed Cell Volume', 'White Blood Cells (cells/cmm)', 'Red Blood Cells (millions/cmm)']
categoricals=[]
for col in data.columns:
    if not col in numeric:
        categoricals.append(col)
categoricals.remove('Chronic Kidney Disease')
```

### Note:

Note that Specific Gravity, Albumin and Sugar basically are categorical features. But because they're ordinal, they will be preprocessed as numerical.

```
numeric
categoricals
```

### Notes:

Some features have high proportions of missing values, thus they cannot be imputed with measures of central tendency. That would distort their distributions.

###Let's take a look at categorical features now.

```
import matplotlib.style as style
style.use('fivethirtyeight')
style.use('seaborn-darkgrid')
n_rows, n_cols = (int(len(categoricals)/2),2) #Since we have 10 categorical features in total
```

#Initializing the subplot

```
figure, axes = plt.subplots(nrows=n_rows, ncols=n_cols,figsize=(20, 30))
figure.suptitle('\n\nCountplots of Categorical Features',fontsize=16)
for index, column in enumerate(categoricals):
```

#The "coordinates" of each plot

i,j = index // n_cols, index % n_cols

#Calculating the missing values percentage

miss_perc="%.2f"%(100*(1-(data[column].dropna().shape[0])/data.shape[0]))

collabel=column+"\n({}% is missing)".format(miss_perc)

#Visualising the count of the categorical features

fig = sns.countplot(x=column, data=data,label=collabel, palette=sns.cubehelix_palette(rot=-.35,light=0.85,hue=1),ax= axes[i,j])

axes[i,j].set_title(collabel,fontsize=12)

axes[i,j].set_xlabel(None)

axes[i,j].set_ylabel("Count")

axes[i,j].set_xticklabels(axes[i,j].get_xticklabels(),fontsize=10)

plt.show()

### Notes:

Some features have very high percentages of missing values while some have almost none. this sample of 400 people may not well-represent the population of India (which is where the data was collected), nor any other population.

#Calculating the missing values percentage of the target

miss_perc="%.2f"%(100*(1-(data['Chronic Kidney Disease'].dropna().shape[0])/data.shape[0]))

label="Disease\n(missing:\n{}%)".format(miss_perc)

#Visualising the count of the target

fig=sns.countplot(x=data['Chronic Kidney Disease'],label=label, palette=sns.cubehelix_palette(rot=-.35,light=0.85,hue=1))

plt.title("Disease\n({}% is missing)".format(miss_perc),fontsize=14)

plt.show()

### Notes:We can see there are 0 missing values in the target, which is logical because this is not a semi-supervised problem.

###This further shows that this dataset represents neither a country/community nor even a group of ill people (more than 60% of the population is ill).

#Let's take a look at missing values.

style.use('seaborn-darkgrid')

d=((data.isnull().sum()/data.shape[0])).sort_values(ascending=False)

d.plot(kind='bar',

    color=sns.cubehelix_palette(start=2,

```
                        rot=0.15,
                        dark=0.15,
                        light=0.95,
                        reverse=True,
                        n_colors=24),
      figsize=(20,10))
plt.title("\nProportions of Missing Values:\n",fontsize=40)
plt.show()
```

###Finally, we can conclude that the transformer that best kept the distribution of the data is the wide robust scaler, which is pretty logical since we have a few bust serious outliers

# Exploratory Data Analysis

```
from sklearn.impute import KNNImputer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
knnimp=KNNImputer(weights='distance', n_neighbors=8)
data_imp = knnimp.fit_transform(data)
style.use('seaborn-darkgrid')
numericdat=data.drop(categoricals, axis=1, inplace=False)
plt.figure(figsize=(15,15))
sns.heatmap(numericdat.corr("pearson"),
      cmap=sns.diverging_palette(280, 280, s=100, l=35, as_cmap=True,sep=60),
      square=True,
      annot=True,
      annot_kws={'fontsize':7},
      fmt='.2%',
      cbar=False)
plt.title("Pearson Correlation Matrix\n",fontsize=9)
plt.show()
```

### Notes:We can notice high correlations(>70% and >-70%) between certain columns.

```
n_rows, n_cols = (5,2)
figure, axes = plt.subplots(nrows=n_rows, ncols=n_cols,figsize=(30, 40))
```

```
figure.suptitle('\n\nCategorical Features\nVS\nTarget Variable',fontsize=20)
for index, column in enumerate(categoricals):
    i,j = (index // n_cols), (index % n_cols)
    sns.heatmap(pd.crosstab(data[column],data['Chronic Kidney Disease']),
            ax=axes[i,j],
            cmap=sns.cubehelix_palette(start=2.8, rot=.1),
            square='True',
            cbar=False,
            annot=True,
            fmt='d')
    axes[i,j].set_xlabel("Disease")
    axes[i,j].set_ylabel(column)
    axes[i,j].set_yticklabels(axes[i,j].get_yticklabels())
    axes[i,j].set_xticklabels(["No CKD","CKD"])
plt.show()
```

#Prediction

```
X=data_imp[:,:24]
Y=data_imp[:,24]
WRS=RobustScaler(quantile_range=(15,85))
scaled_data=WRS.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(scaled_data, Y, test_size=0.2, random_state=12)
```

### Linear-Kernel SVC The distributions shown in the next graph represent LDA results only on training data

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
lin_svc=SVC(kernel='linear')
n_rows, n_cols= 6,4
figure, axes = plt.subplots(nrows=n_rows,ncols= n_cols, figsize=(20,120))
```

```
figure.suptitle('\nLDA with Linear SVC\n(Distributions represent\nonly the training data)',fontsize=16)
for index in range(24):
    i,j = (index // n_cols), (index % n_cols)
    pca=PCA(n_components=index+1)
    lda=LinearDiscriminantAnalysis()
    some_pipe=make_pipeline(pca,lda)
    X_lda_train=some_pipe.fit_transform(X_train,Y_train)
    X_lda_test=some_pipe.fit_transform(X_test,Y_test)
    lin_svc.fit(X_lda_train,Y_train)
    y_pred_train=lin_svc.predict(X_lda_train)
    train_acc= accuracy_score(y_pred_train,Y_train)
    y_pred_test=lin_svc.predict(X_lda_test)
    test_acc= accuracy_score(y_pred_test,Y_test)
    X_lda_train=X_lda_train.reshape((320,))
    bp=sns.boxenplot(y=X_lda_train, x=Y_train, color="paleturquoise",showfliers=True,ax=axes[i,j])
    axes[i,j].set_title("n° Of PCA Components: {}\nTraining Accuracy: {}\nTesting Accuracy: {}".format(index+10,"%.3f"%train_acc,"%.3f"%test_acc),fontsize=5)
    axes[i,j].set_xlabel(None)
    axes[i,j].set_xticklabels(["CKD","No CKD"],fontsize=6)
plt.show()
```

###Low variance despite the fact that the testing set is 0.2 of the whole dataset, which only has 400 samples.### Evaluating A Few Other Models

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
SVM_RBF=SVC()
SVM_Poly3=SVC(kernel='poly',degree=3)
KNN8=KNeighborsClassifier(n_neighbors=8,weights='distance')
Naive_Bayes=GaussianNB()
```

```
LogReg=LogisticRegression()
Tree=DecisionTreeClassifier()
Forest=RandomForestClassifier()
models=[SVM_RBF,SVM_Poly3,KNN8,Naive_Bayes,LogReg,Tree,Forest]
names=["SVM_RBF","SVM_Poly3","Weighted 8NearestNeighbors","Naive Bayes","Logistic
Regression","Decision Tree","Random Forest"]
n_rows, n_cols= 7,1
figure,axes = plt.subplots(nrows=n_rows,ncols= n_cols, figsize=(20, 20))
figure.suptitle('\nEvaluating Different Models', fontsize=9)
tr_mask = np.empty(shape=(24,1),dtype="object")
ts_mask = np.empty(shape=(24,1),dtype="object")
for i in range(24):
    tr_mask[i]="Training"
    ts_mask[i]="Testing"
mask = np.vstack((tr_mask,ts_mask))
mask=mask.reshape((48,))
cmps=[i for i in range(1,25)] * 2
for index in range(len(models)):
    pca_tr_acc=[]
    pca_ts_acc=[]
    for n_comps in range(1,25):
        model=models[index]
        pca=PCA(n_components=n_comps)
        pca_model=make_pipeline(pca,model)
        pca_model.fit(X_train,Y_train)
        y_tr_pred= pca_model.predict(X_train)
        pca_tr_acc.append(accuracy_score(y_tr_pred,Y_train))
        y_ts_pred=pca_model.predict(X_test)
        pca_ts_acc.append(accuracy_score(y_ts_pred,Y_test))
    model_data = pd.DataFrame()
    model_data["PCA"] = pca_tr_acc + pca_ts_acc
    model_data["Results"] = mask
```

```
sns.barplot(x=cmps, y="PCA", hue="Results", data=model_data, palette='cool', ax=axes[index]).set(ylim=(0.8,1))
    axes[index].set_title(names[index],fontsize=40)
    axes[index].set_xlabel("n° of PCA Components",fontsize=15)
    axes[index].set_ylabel("Accuracy",fontsize=35)
    axes[index].set_xticklabels(axes[index].get_xticklabels(),fontsize=15)
plt.show()
from sklearn.metrics import accuracy_score
base_models = [
    SVC(),
    SVC(degree=2, kernel='poly'),
    SVC(kernel='poly'),
    GaussianNB(),
    LogisticRegression(),
    DecisionTreeClassifier(),
    RandomForestClassifier()
]
for mod in base_models:
    mod.fit(X_train, Y_train)
    y_pred = mod.predict(X_test)
    accuracy = accuracy_score(Y_test, y_pred)
    print(mod, accuracy)
```

###As we can see the models didn't really improve comparing to the ones without boosting Neural Networks So We will be trying a small neural network with one hidden layer containing 4 neurons, and a bigger one with 3 hidden layers and lots of neurons.

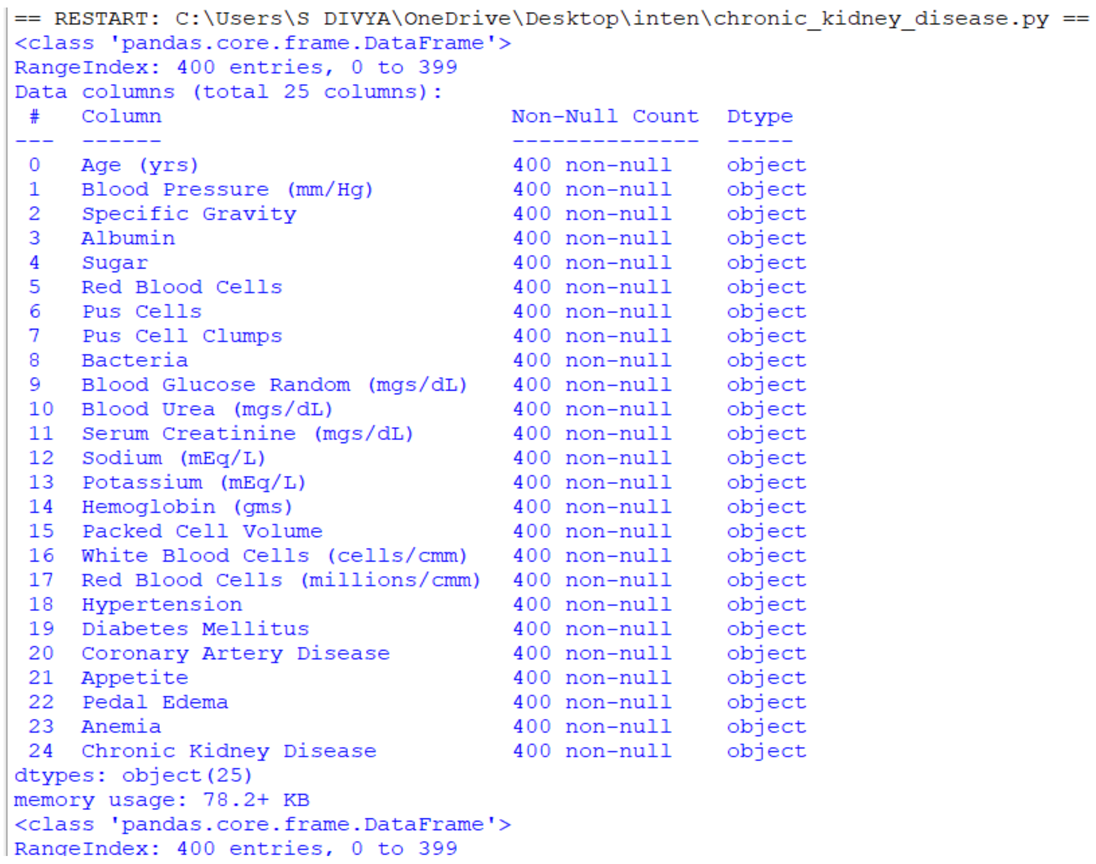
```
import tensorflow
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import to_categorical
early_stopping_monitor = EarlyStopping(patience=5, monitor='accuracy')
Y_net = to_categorical(Y)
pca_tr_acc_1=[]
```

```
pca_ts_acc_1=[]
pca_tr_acc_2=[]
pca_ts_acc_2=[]
for i in range(1,25):
    pca=PCA(n_components=i)
    X_pca=pca.fit_transform(scaled_data)
    X_pca_train, X_pca_test, Y_train, Y_test = train_test_split(X_pca, Y_net, test_size=0.25, random_state=12)
 #little net
   net1= Sequential()
   net1.add(Dense(4, activation='relu', input_shape = (i,)))
   net1.add(Dense(2, activation='softmax'))
   net1.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
   net1.fit(X_pca_train, Y_train, epochs=50, callbacks=[early_stopping_monitor], verbose=0)
   y_tr_pred=net1.predict(X_pca_train)
   pca_tr_acc_1.append(accuracy_score(np.argmax(y_tr_pred, axis=1),Y_train[:,1]))
   y_ts_pred=net1.predict(X_pca_test)
   pca_ts_acc_1.append(accuracy_score(np.argmax(y_ts_pred, axis=1),Y_test[:,1]))
    #big net
    net2= Sequential()
    net2.add(Dense(50, activation='relu', input_shape = (i,)))
    net2.add(Dense(30, activation='relu'))
    net2.add(Dense(20, activation='relu'))
    net2.add(Dense(10, activation='relu'))
    net2.add(Dense(2, activation='softmax'))
   net2.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
   net2.fit(X_pca_train, Y_train, epochs=100, callbacks=[early_stopping_monitor], verbose=0)
   y_tr_pred=net2.predict(X_pca_train)
   pca_tr_acc_2.append(accuracy_score(np.argmax(y_tr_pred, axis=1),Y_train[:,1]))
   y_ts_pred=net2.predict(X_pca_test)
   pca_ts_acc_2.append(accuracy_score(np.argmax(y_ts_pred, axis=1),Y_test[:,1]))
tr_mask = np.empty(shape=(24,1),dtype="object")
```

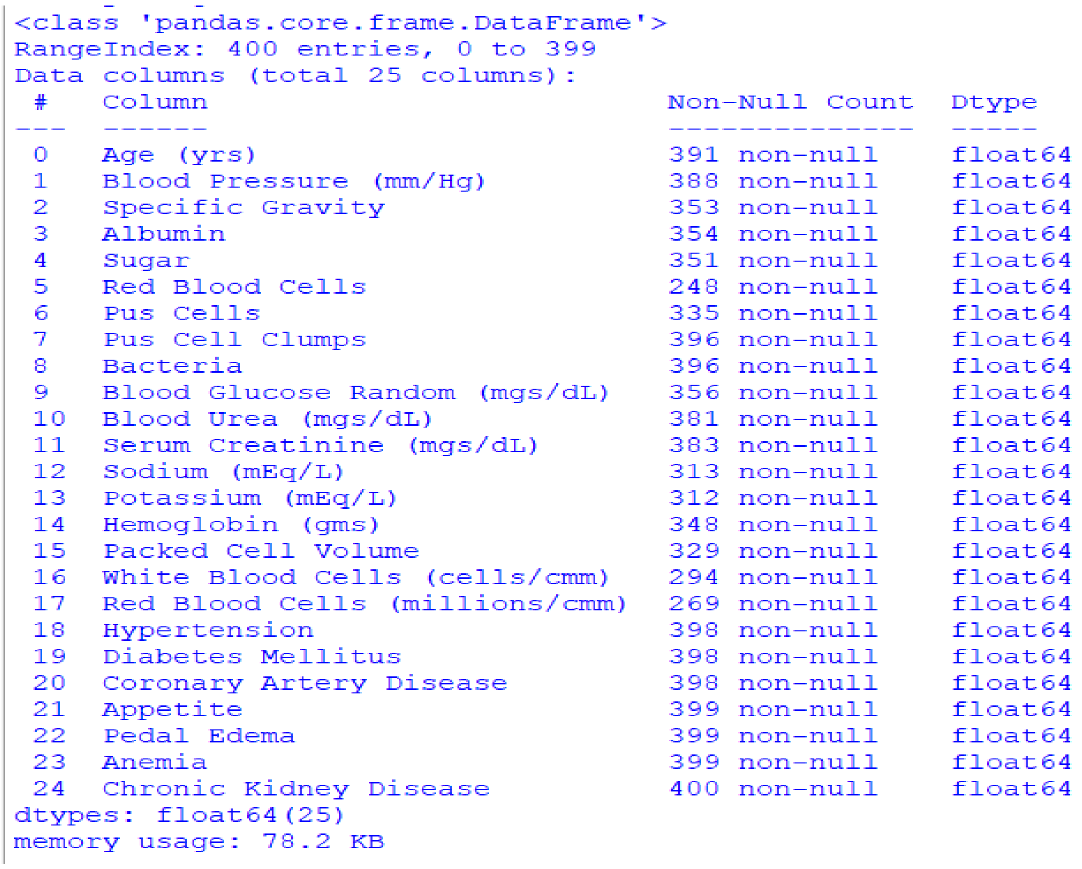
```
ts_mask = np.empty(shape=(24,1),dtype="object")
for i in range(24):
    tr_mask[i]="Training"
    ts_mask[i]="Testing"
mask = np.vstack((tr_mask,ts_mask))
mask=mask.reshape((48,))
cmps=[i for i in range(1,25)] * 2
net1_data = pd.DataFrame()
net1_data["PCA"] = pca_tr_acc_1 + pca_ts_acc_1
net1_data["Results"] = mask
net2_data = pd.DataFrame()
net2_data["PCA"] = pca_tr_acc_2 + pca_ts_acc_2
net2_data["Results"] = mask
n_rows, n_cols= 1,2
figure, axes = plt.subplots(nrows=n_rows,ncols= n_cols, figsize=(30, 10))
figure.suptitle('Little NN vs Big(ger) NN')
graph1=sns.barplot(x=cmps, y="PCA", hue="Results", data=net1_data, palette='cool', ax=axes[0]).set(ylim=(0.5,1))
axes[0].set_title("Little Neural Network")
axes[0].set_xlabel("n° of PCA Components")
axes[0].set_ylabel("Accuracy")
axes[0].set_xticklabels(axes[0].get_xticklabels())
graph2=sns.barplot(x=cmps, y="PCA", hue="Results", data=net2_data, palette='cool', ax=axes[1]).set(ylim=(0.5,1))
axes[1].set_title("Big(ger) Neural Network")
axes[1].set_xlabel("n° of PCA Components")
axes[1].set_ylabel("Accuracy")
axes[1].set_xticklabels(axes[1].get_xticklabels())
plt.show()
```

7.Results

7.1 Output Screens

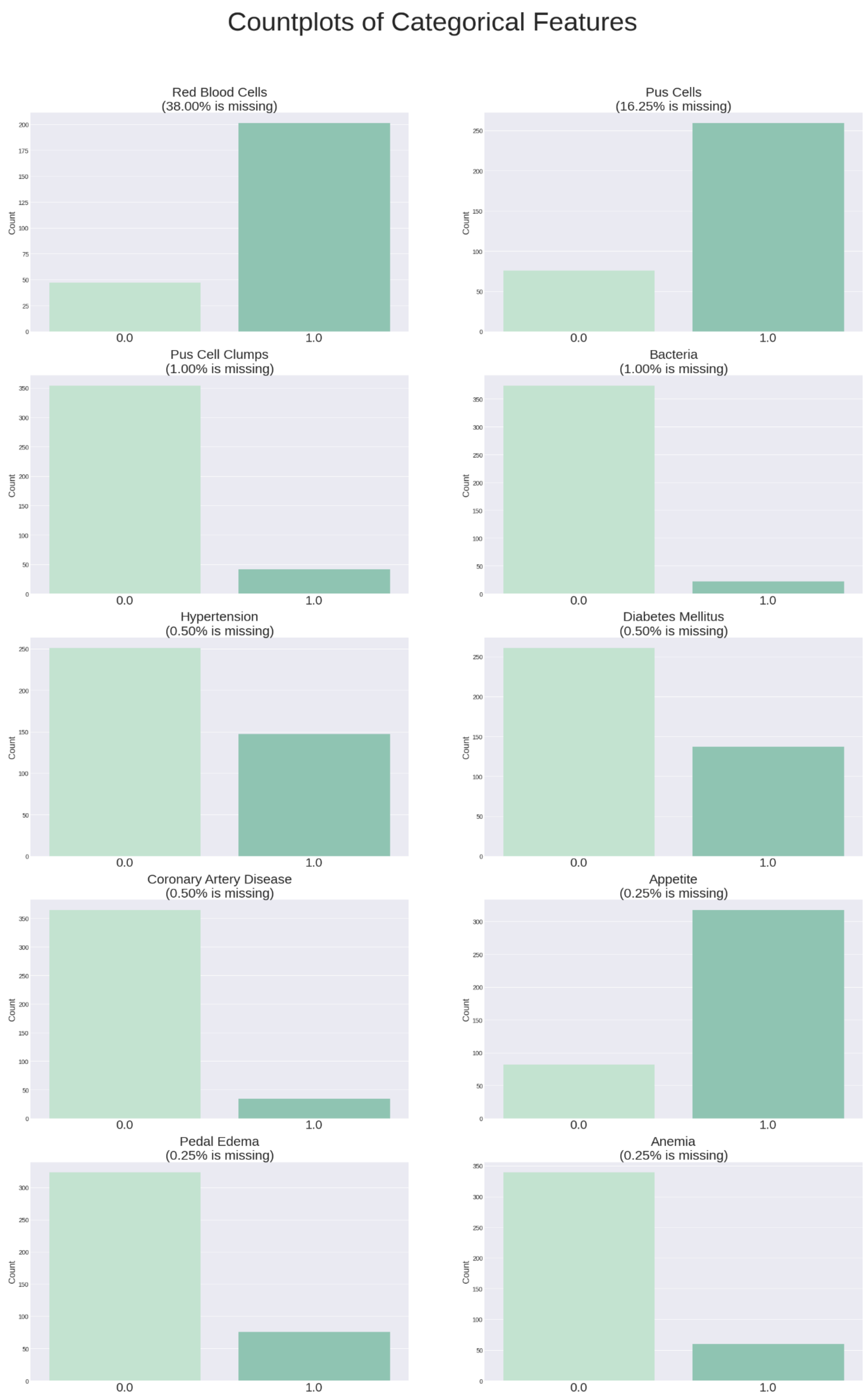


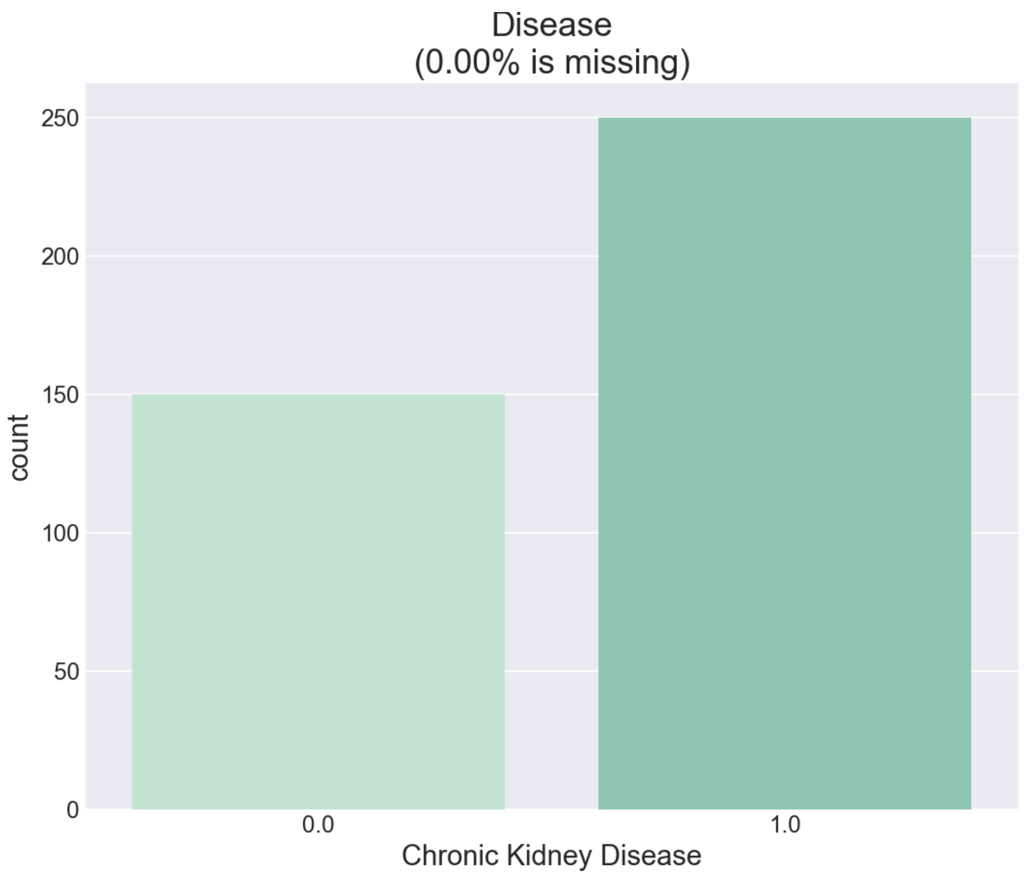
```
== RESTART: C:\Users\S DIVYA\OneDrive\Desktop\inten\chronic_kidney_disease.py ==
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   Age (yrs)                     400 non-null    object
 1   Blood Pressure (mm/Hg)        400 non-null    object
 2   Specific Gravity              400 non-null    object
 3   Albumin                       400 non-null    object
 4   Sugar                         400 non-null    object
 5   Red Blood Cells               400 non-null    object
 6   Pus Cells                     400 non-null    object
 7   Pus Cell Clumps               400 non-null    object
 8   Bacteria                      400 non-null    object
 9   Blood Glucose Random (mgs/dL) 400 non-null    object
 10  Blood Urea (mgs/dL)           400 non-null    object
 11  Serum Creatinine (mgs/dL)     400 non-null    object
 12  Sodium (mEq/L)                400 non-null    object
 13  Potassium (mEq/L)             400 non-null    object
 14  Hemoglobin (gms)              400 non-null    object
 15  Packed Cell Volume            400 non-null    object
 16  White Blood Cells (cells/cmm) 400 non-null    object
 17  Red Blood Cells (millions/cmm) 400 non-null   object
 18  Hypertension                  400 non-null    object
 19  Diabetes Mellitus             400 non-null    object
 20  Coronary Artery Disease       400 non-null    object
 21  Appetite                      400 non-null    object
 22  Pedal Edema                   400 non-null    object
 23  Anemia                        400 non-null    object
 24  Chronic Kidney Disease        400 non-null    object
dtypes: object(25)
memory usage: 78.2+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Age (yrs)                      391 non-null    float64
 1   Blood Pressure (mm/Hg)         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    float64
 6   Pus Cells                      335 non-null    float64
 7   Pus Cell Clumps                396 non-null    float64
 8   Bacteria                       396 non-null    float64
 9   Blood Glucose Random (mgs/dL)  356 non-null    float64
 10  Blood Urea (mgs/dL)            381 non-null    float64
 11  Serum Creatinine (mgs/dL)      383 non-null    float64
 12  Sodium (mEq/L)                 313 non-null    float64
 13  Potassium (mEq/L)              312 non-null    float64
 14  Hemoglobin (gms)               348 non-null    float64
 15  Packed Cell Volume             329 non-null    float64
 16  White Blood Cells (cells/cmm)  294 non-null    float64
 17  Red Blood Cells (millions/cmm) 269 non-null    float64
 18  Hypertension                   398 non-null    float64
 19  Diabetes Mellitus              398 non-null    float64
 20  Coronary Artery Disease        398 non-null    float64
 21  Appetite                       399 non-null    float64
 22  Pedal Edema                    399 non-null    float64
 23  Anemia                         399 non-null    float64
 24  Chronic Kidney Disease         400 non-null    float64
dtypes: float64(25)
memory usage: 78.2 KB
```

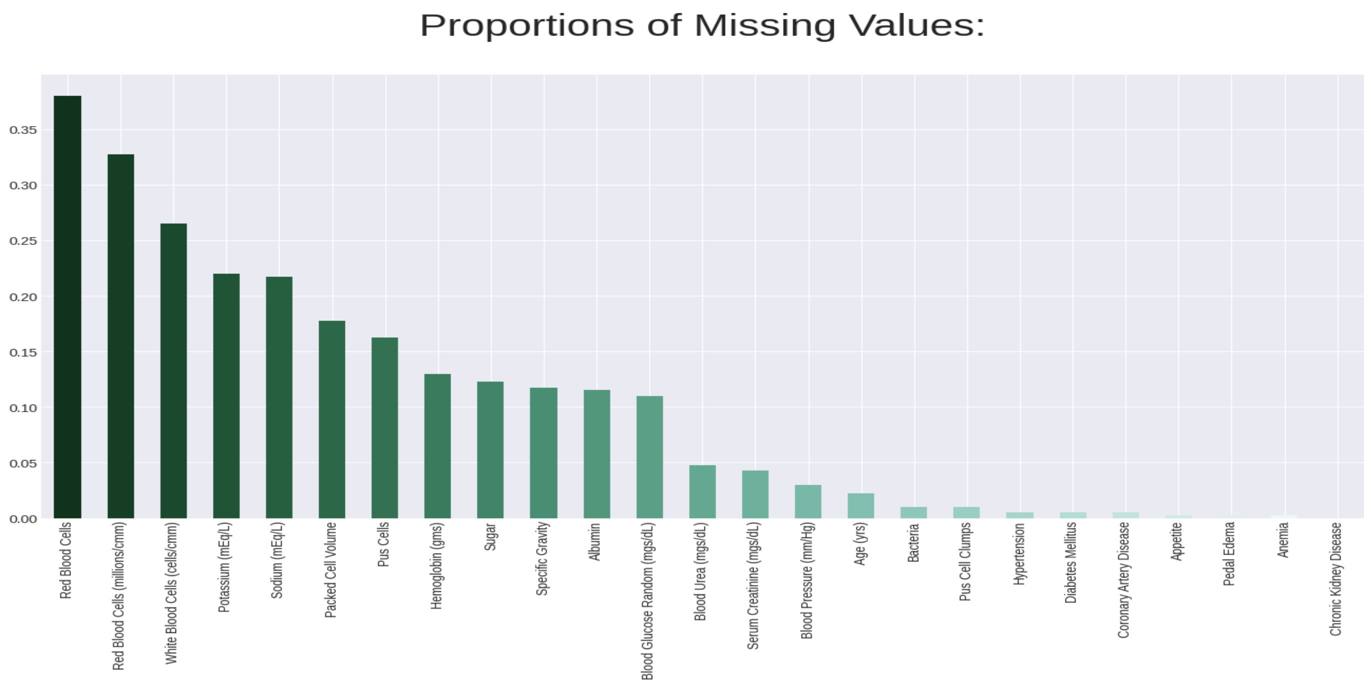
Fig 7.1


Countplots of Categorical Features
Red Blood Cells (38.00% is missing)
Pus Cells (16.25% is missing)
Pus Cell Clumps (1.00% is missing)
Bacteria (1.00% is missing)
Hypertension (0.50% is missing)
Diabetes Mellitus (0.50% is missing)
Coronary Artery Disease (0.50% is missing)
Appetite (0.25% is missing)
Pedal Edema (0.25% is missing)
Anemia (0.25% is missing)
Count
0.0
1.0



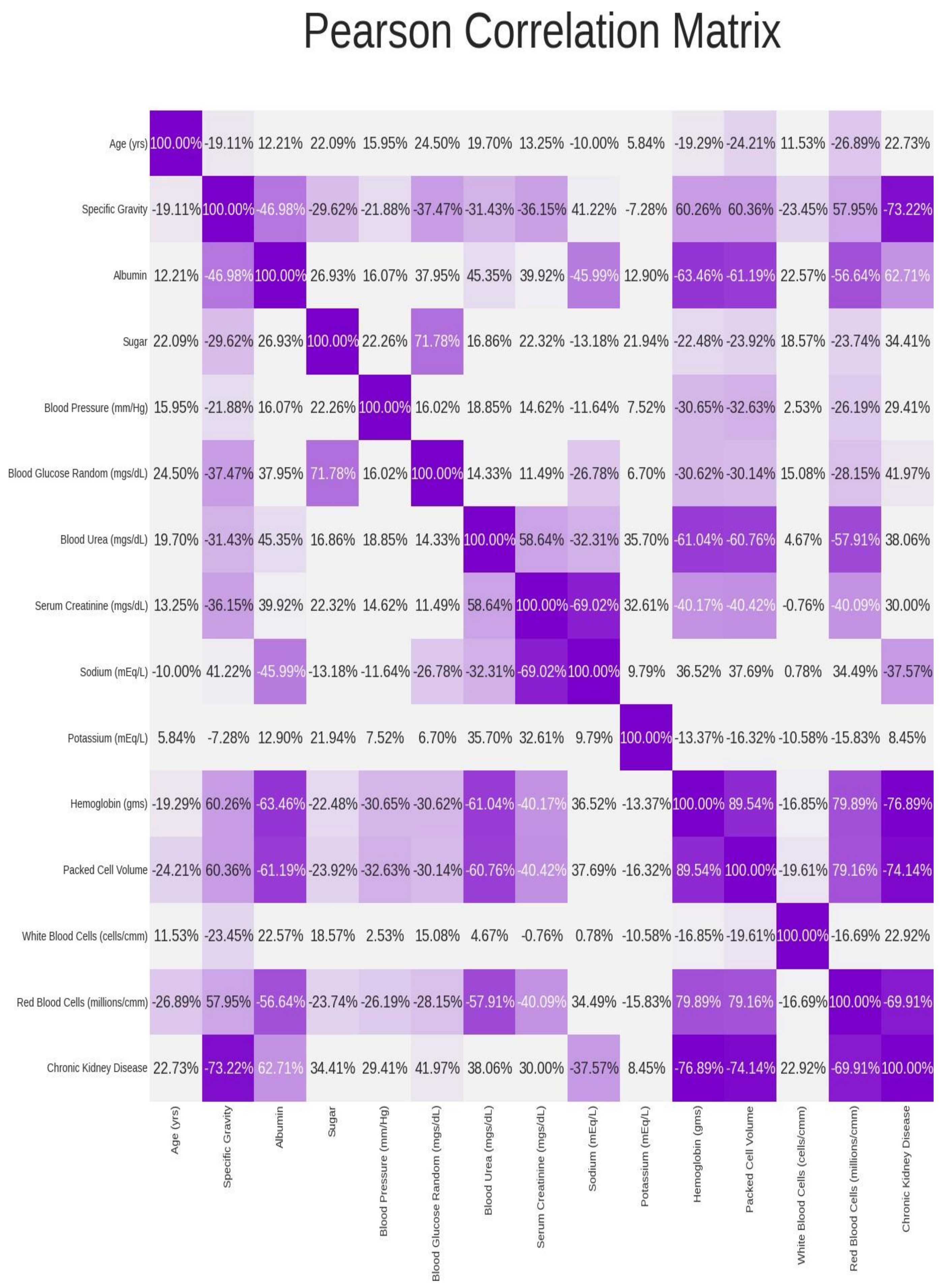
Disease
(0.00% is missing)
count
250
200
150
100
50
0
0.0
1.0
Chronic Kidney Disease

Fig 7.2



Proportions of Missing Values:
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Red Blood Cells
Red Blood Cells (millions/cmm)
White Blood Cells (cells/cmm)
Potassium (mEq/L)
Sodium (mEq/L)
Packed Cell Volume
Pus Cells
Hemoglobin (gms)
Sugar
Specific Gravity
Albumin
Blood Glucose Random (mgs/dL)
Blood Urea (mgs/dL)
Serum Creatinine (mgs/dL)
Blood Pressure (mm/Hg)
Age (yrs)
Bacteria
Pus Cell Clumps
Hypertension
Diabetes Mellitus
Coronary Artery Disease
Appetite
Pedal Edema
Anemia
Chronic Kidney Disease

Fig 7.3



	Age (yrs)	Specific Gravity	Albumin	Sugar	Blood Pressure (mm/Hg)	Blood Glucose Random (mgs/dL)	Blood Urea (mgs/dL)	Serum Creatinine (mgs/dL)	Sodium (mEq/L)	Potassium (mEq/L)	Hemoglobin (gms)	Packed Cell Volume	White Blood Cells (cells/cmm)	Red Blood Cells (millions/cmm)	Chronic Kidney Disease
Age (yrs)	100.00%	-19.11%	12.21%	22.09%	15.95%	24.50%	19.70%	13.25%	-10.00%	5.84%	-19.29%	-24.21%	11.53%	-26.89%	22.73%
Specific Gravity	-19.11%	100.00%	-46.98%	-29.62%	-21.88%	-37.47%	-31.43%	-36.15%	41.22%	-7.28%	60.26%	60.36%	-23.45%	57.95%	-73.22%
Albumin	12.21%	-46.98%	100.00%	26.93%	16.07%	37.95%	45.35%	39.92%	-45.99%	12.90%	-63.46%	-61.19%	22.57%	-56.64%	62.71%
Sugar	22.09%	-29.62%	26.93%	100.00%	22.26%	71.78%	16.86%	22.32%	-13.18%	21.94%	-22.48%	-23.92%	18.57%	-23.74%	34.41%
Blood Pressure (mm/Hg)	15.95%	-21.88%	16.07%	22.26%	100.00%	16.02%	18.85%	14.62%	-11.64%	7.52%	-30.65%	-32.63%	2.53%	-26.19%	29.41%
Blood Glucose Random (mgs/dL)	24.50%	-37.47%	37.95%	71.78%	16.02%	100.00%	14.33%	11.49%	-26.78%	6.70%	-30.62%	-30.14%	15.08%	-28.15%	41.97%
Blood Urea (mgs/dL)	19.70%	-31.43%	45.35%	16.86%	18.85%	14.33%	100.00%	58.64%	-32.31%	35.70%	-61.04%	-60.76%	4.67%	-57.91%	38.06%
Serum Creatinine (mgs/dL)	13.25%	-36.15%	39.92%	22.32%	14.62%	11.49%	58.64%	100.00%	-69.02%	32.61%	-40.17%	-40.42%	-0.76%	-40.09%	30.00%
Sodium (mEq/L)	-10.00%	41.22%	-45.99%	-13.18%	-11.64%	-26.78%	-32.31%	-69.02%	100.00%	9.79%	36.52%	37.69%	0.78%	34.49%	-37.57%
Potassium (mEq/L)	5.84%	-7.28%	12.90%	21.94%	7.52%	6.70%	35.70%	32.61%	9.79%	100.00%	-13.37%	-16.32%	-10.58%	-15.83%	8.45%
Hemoglobin (gms)	-19.29%	60.26%	-63.46%	-22.48%	-30.65%	-30.62%	-61.04%	-40.17%	36.52%	-13.37%	100.00%	89.54%	-16.85%	79.89%	-76.89%
Packed Cell Volume	-24.21%	60.36%	-61.19%	-23.92%	-32.63%	-30.14%	-60.76%	-40.42%	37.69%	-16.32%	89.54%	100.00%	-19.61%	79.16%	-74.14%
White Blood Cells (cells/cmm)	11.53%	-23.45%	22.57%	18.57%	2.53%	15.08%	4.67%	-0.76%	0.78%	-10.58%	-16.85%	-19.61%	100.00%	-16.69%	22.92%
Red Blood Cells (millions/cmm)	-26.89%	57.95%	-56.64%	-23.74%	-26.19%	-28.15%	-57.91%	-40.09%	34.49%	-15.83%	79.89%	79.16%	-16.69%	100.00%	-69.91%
Chronic Kidney Disease	22.73%	-73.22%	62.71%	34.41%	29.41%	41.97%	38.06%	30.00%	-37.57%	8.45%	-76.89%	-74.14%	22.92%	-69.91%	100.00%

Fig 7.4

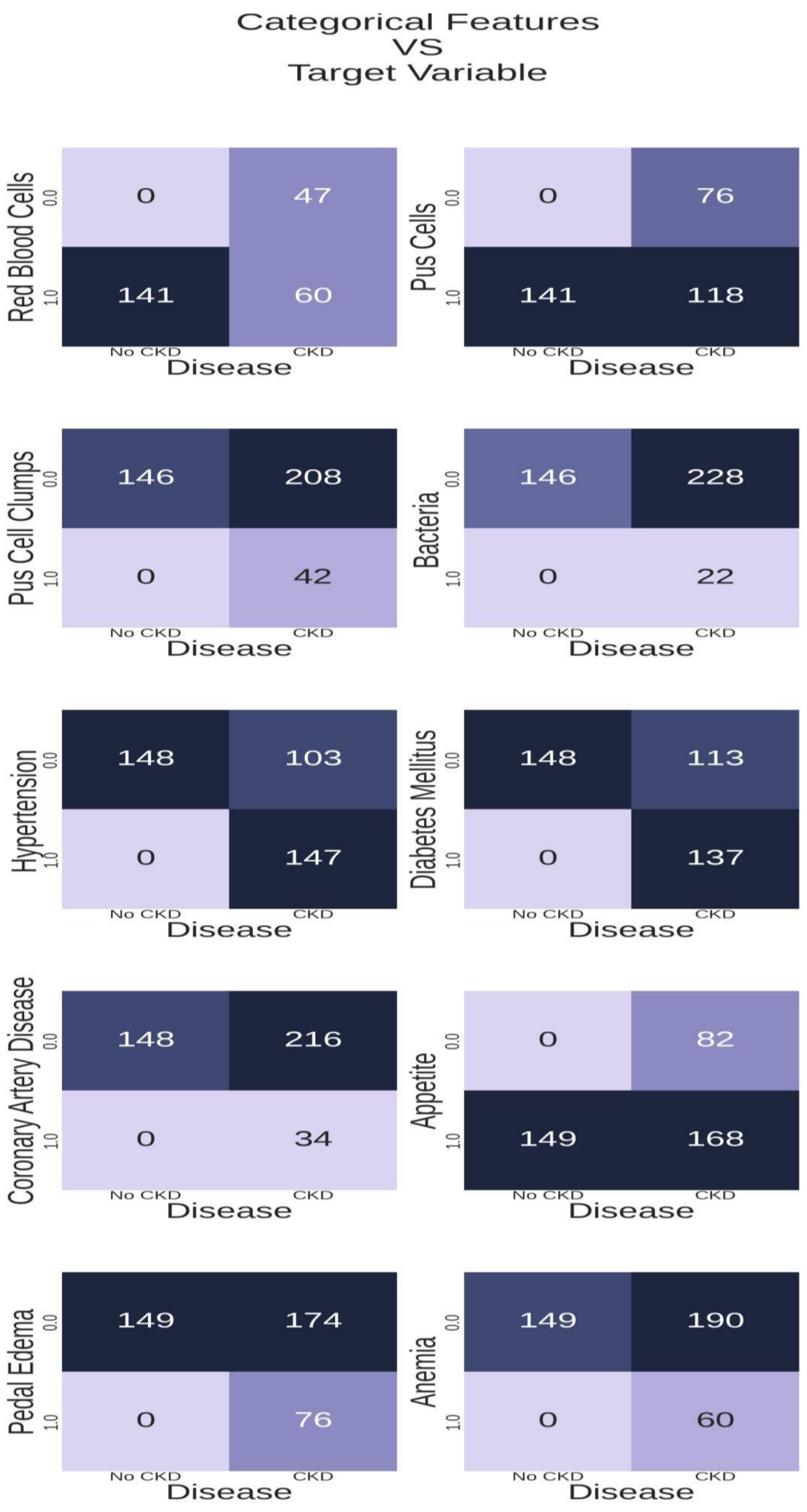
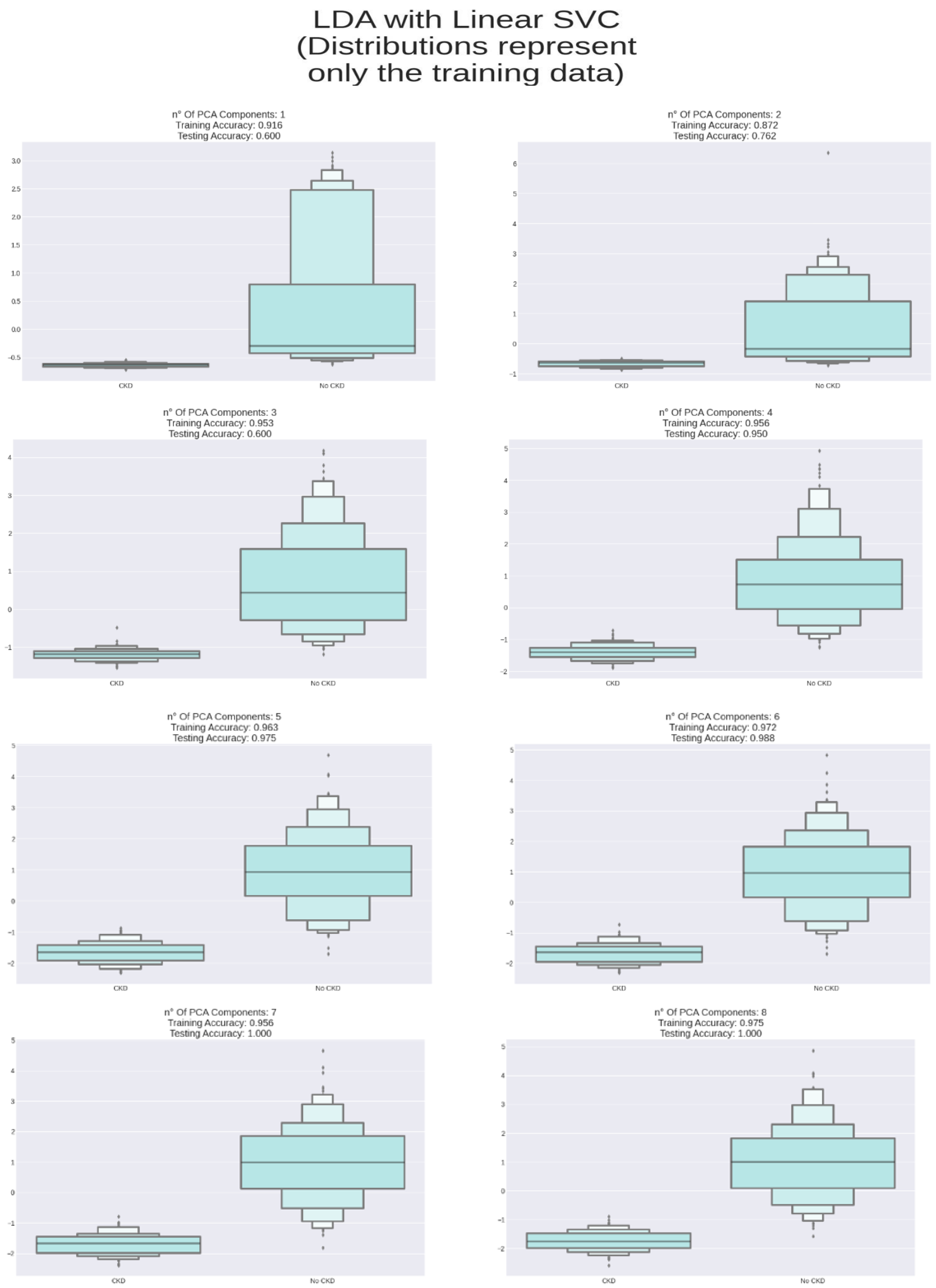
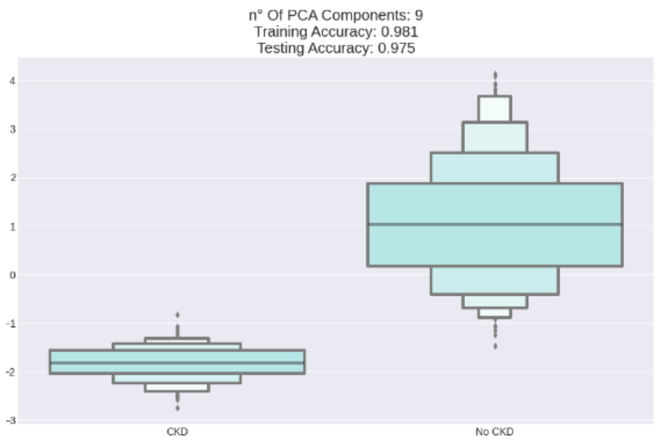
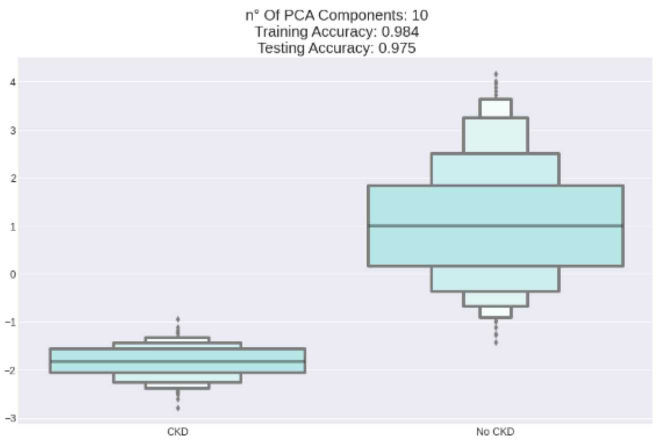

Categorical Features
VS
Target Variable
Red Blood Cells
0.0
1.0
0
47
141
60
No CKD
CKD
Disease
Pus Cells
0.0
1.0
0
76
141
118
No CKD
CKD
Disease
Pus Cell Clumps
0.0
1.0
146
208
0
42
No CKD
CKD
Disease
Bacteria
0.0
1.0
146
228
0
22
No CKD
CKD
Disease
Hypertension
0.0
1.0
148
103
0
147
No CKD
CKD
Disease
Diabetes Mellitus
0.0
1.0
148
113
0
137
No CKD
CKD
Disease
Coronary Artery Disease
0.0
1.0
148
216
0
34
No CKD
CKD
Disease
Appetite
0.0
1.0
0
82
149
168
No CKD
CKD
Disease
Pedal Edema
0.0
1.0
149
174
0
76
No CKD
CKD
Disease
Anemia
0.0
1.0
149
190
0
60
No CKD
CKD
Disease

Fig 7.5

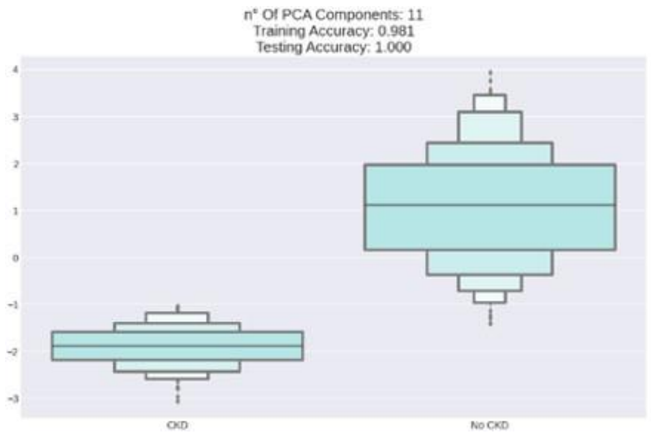

LDA with Linear SVC
(Distributions represent
only the training data)
n° Of PCA Components: 1
Training Accuracy: 0.916
Testing Accuracy: 0.600
n° Of PCA Components: 2
Training Accuracy: 0.872
Testing Accuracy: 0.762
n° Of PCA Components: 3
Training Accuracy: 0.953
Testing Accuracy: 0.600
n° Of PCA Components: 4
Training Accuracy: 0.956
Testing Accuracy: 0.950
n° Of PCA Components: 5
Training Accuracy: 0.963
Testing Accuracy: 0.975
n° Of PCA Components: 6
Training Accuracy: 0.972
Testing Accuracy: 0.988
n° Of PCA Components: 7
Training Accuracy: 0.956
Testing Accuracy: 1.000
n° Of PCA Components: 8
Training Accuracy: 0.975
Testing Accuracy: 1.000
CKD
No CKD



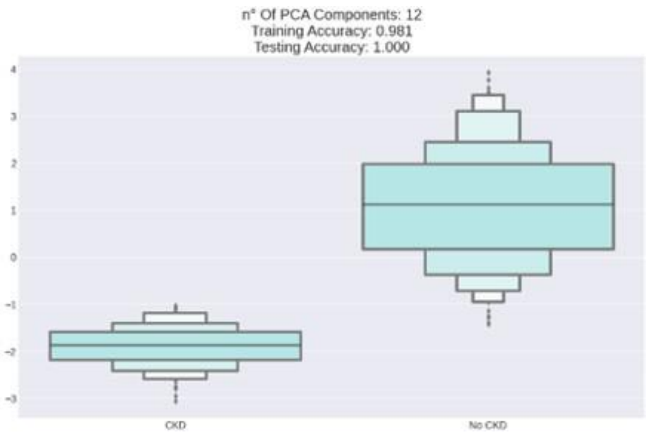
n° Of PCA Components: 9
Training Accuracy: 0.981
Testing Accuracy: 0.975
CKD
No CKD



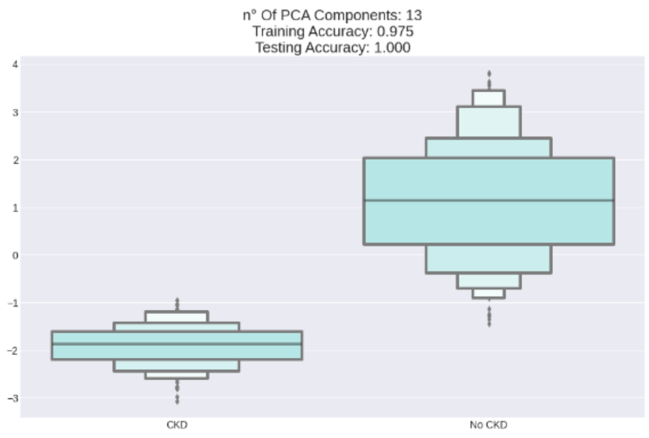
n° Of PCA Components: 10
Training Accuracy: 0.984
Testing Accuracy: 0.975
CKD
No CKD



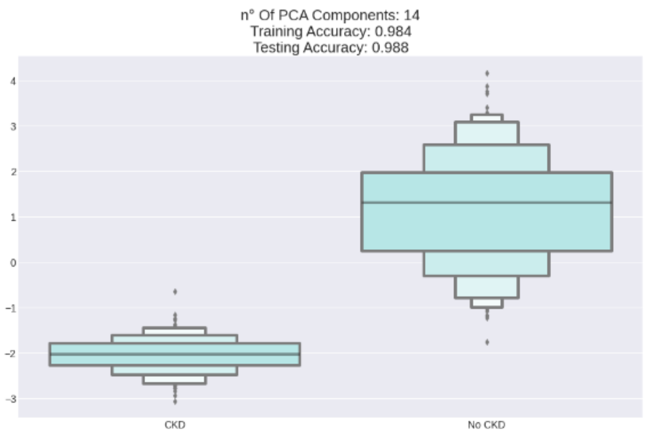
n° Of PCA Components: 11
Training Accuracy: 0.981
Testing Accuracy: 1.000
CKD
No CKD



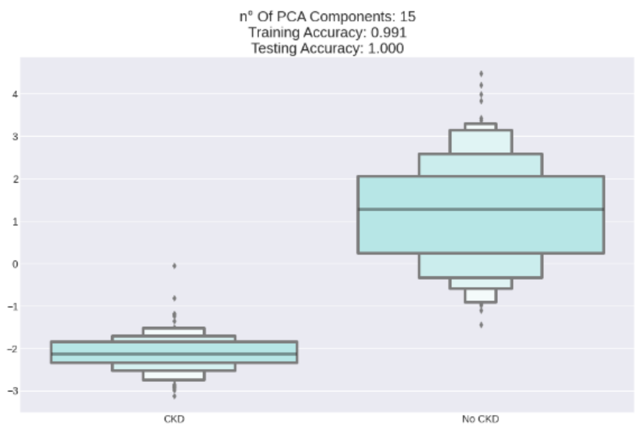
n° Of PCA Components: 12
Training Accuracy: 0.981
Testing Accuracy: 1.000
CKD
No CKD



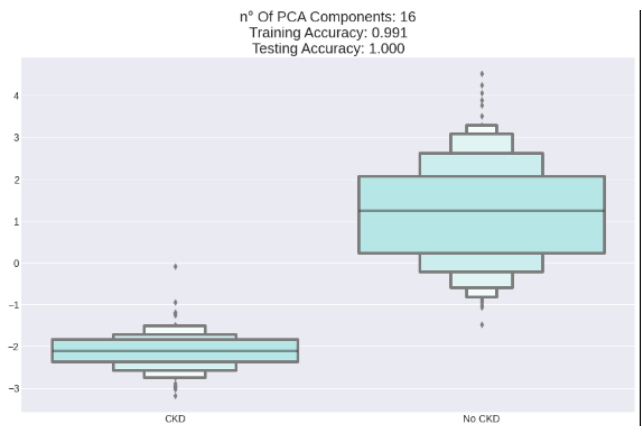
n° Of PCA Components: 13
Training Accuracy: 0.975
Testing Accuracy: 1.000
CKD
No CKD



n° Of PCA Components: 14
Training Accuracy: 0.984
Testing Accuracy: 0.988
CKD
No CKD



n° Of PCA Components: 15
Training Accuracy: 0.991
Testing Accuracy: 1.000
CKD
No CKD



n° Of PCA Components: 16
Training Accuracy: 0.991
Testing Accuracy: 1.000
CKD
No CKD

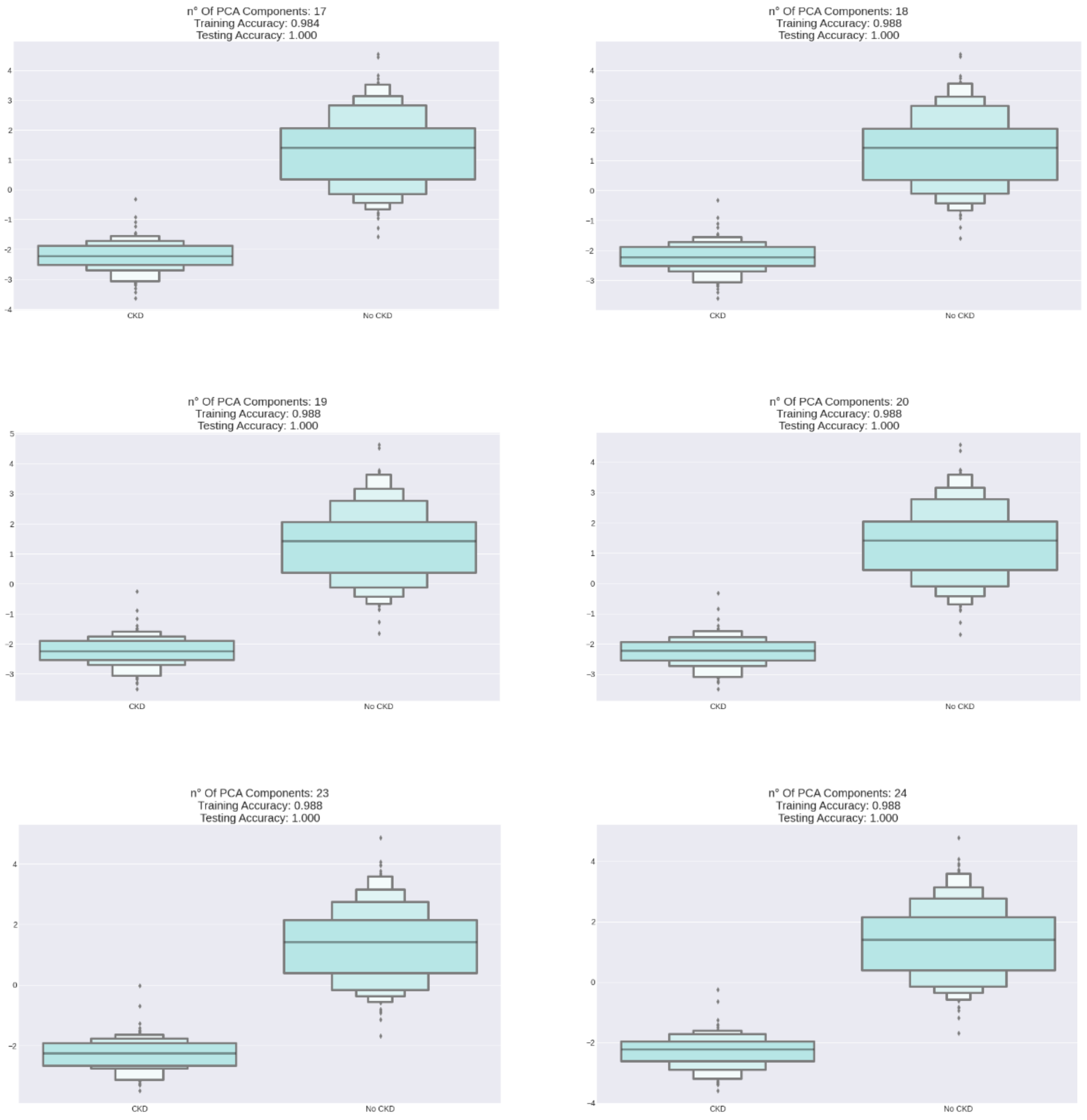
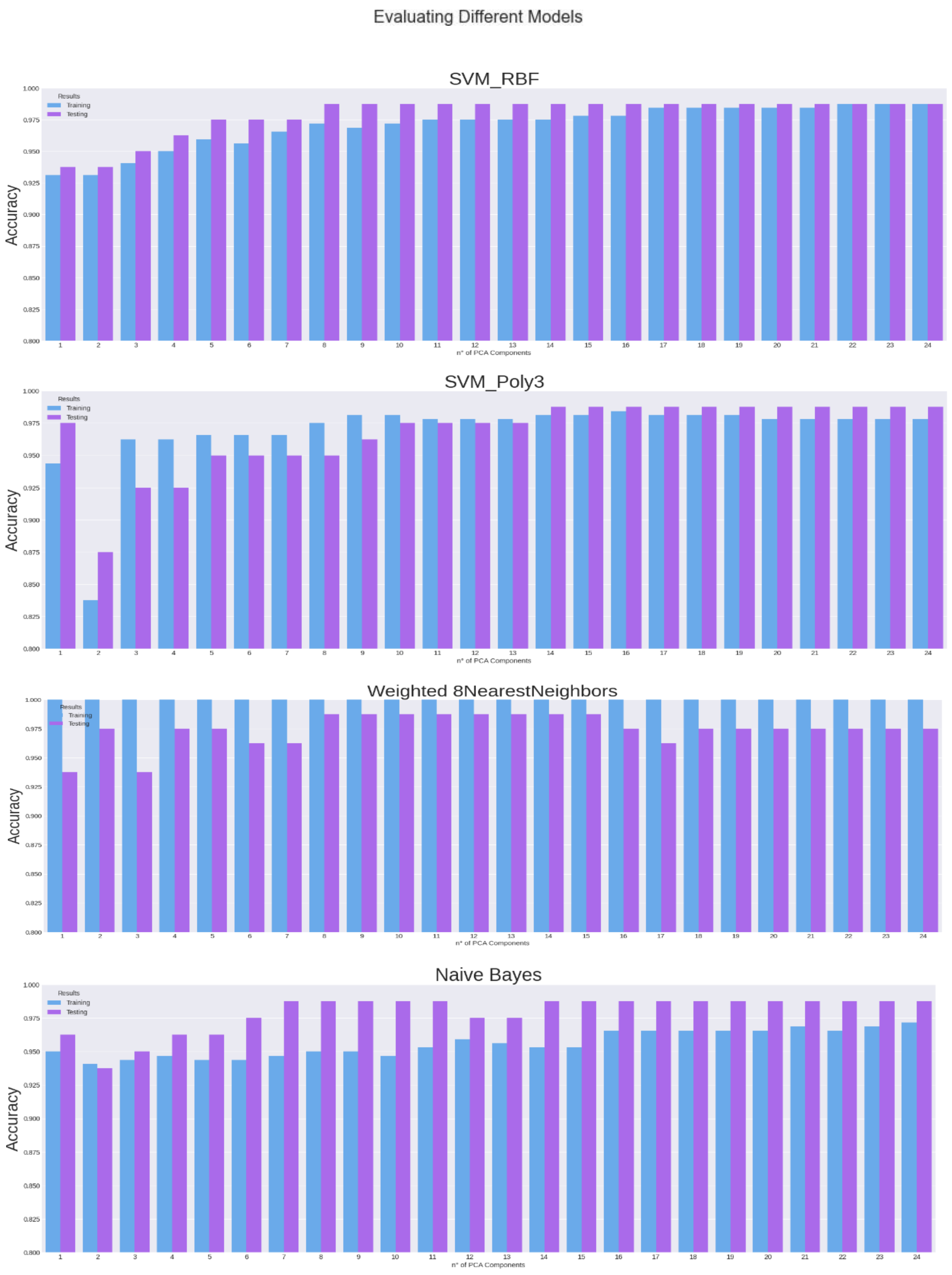

n° Of PCA Components: 17
Training Accuracy: 0.984
Testing Accuracy: 1.000
n° Of PCA Components: 18
Training Accuracy: 0.988
Testing Accuracy: 1.000
n° Of PCA Components: 19
Training Accuracy: 0.988
Testing Accuracy: 1.000
n° Of PCA Components: 20
Training Accuracy: 0.988
Testing Accuracy: 1.000
n° Of PCA Components: 23
Training Accuracy: 0.988
Testing Accuracy: 1.000
n° Of PCA Components: 24
Training Accuracy: 0.988
Testing Accuracy: 1.000
CKD
No CKD

Fig 7.6


Evaluating Different Models
SVM_RBF
SVM_Poly3
Weighted 8NearestNeighbors
Naive Bayes
Results
Training
Testing
Accuracy
n° of PCA Components

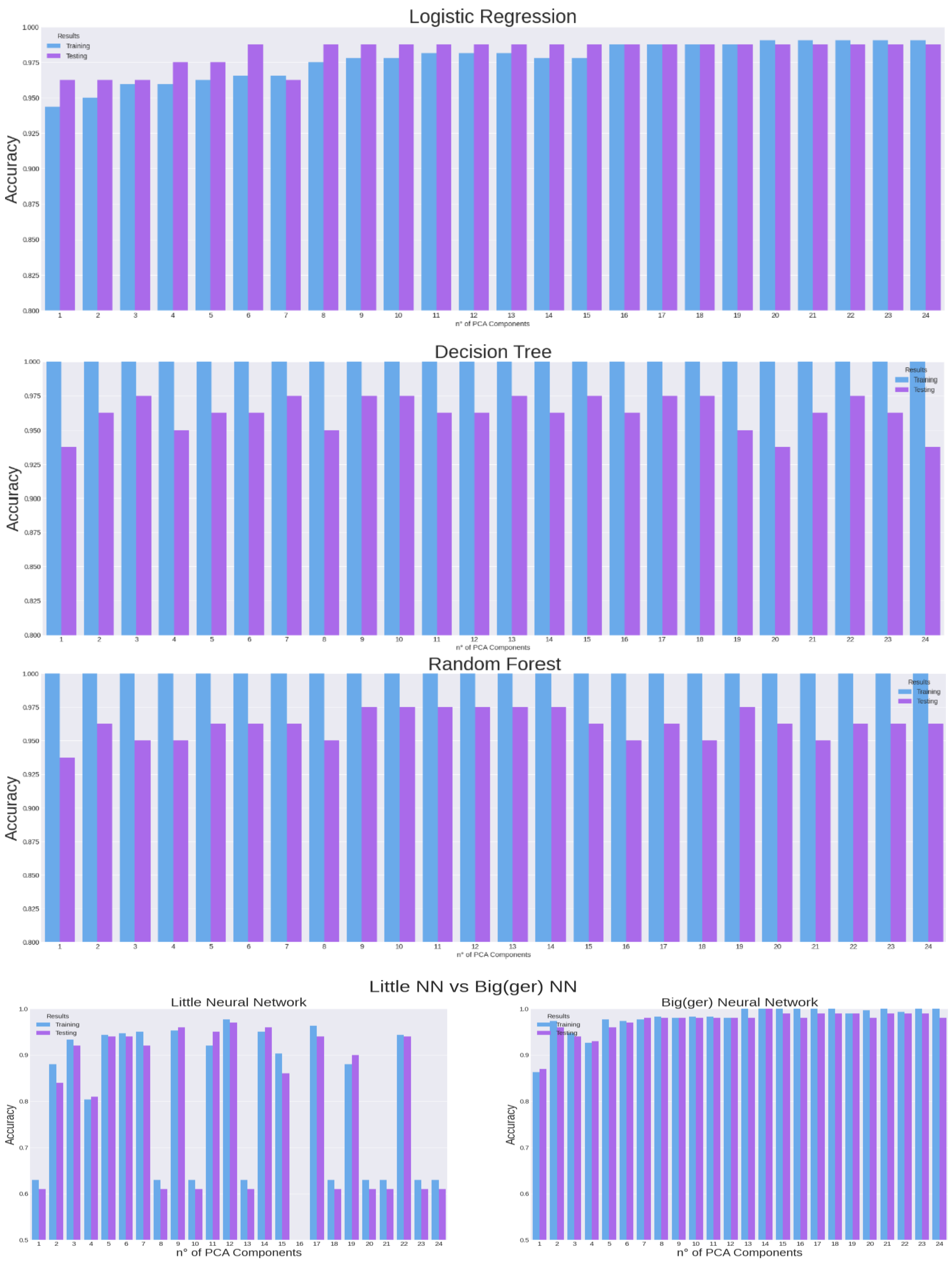

Logistic Regression
Decision Tree
Random Forest
Little NN vs Big(ger) NN
Little Neural Network
Big(ger) Neural Network
Accuracy
n° of PCA Components
Results
Training
Testing

Fig 7.7

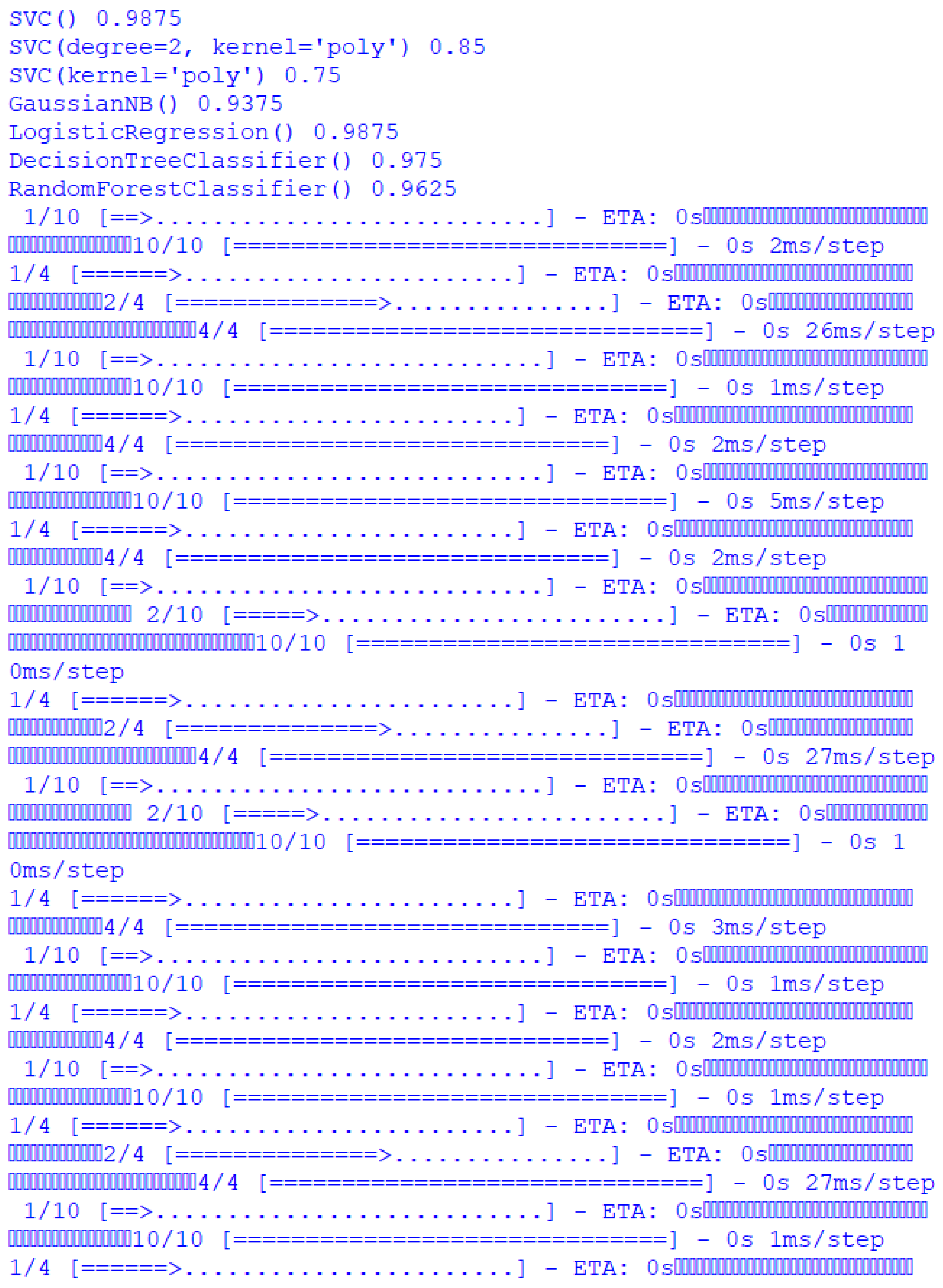
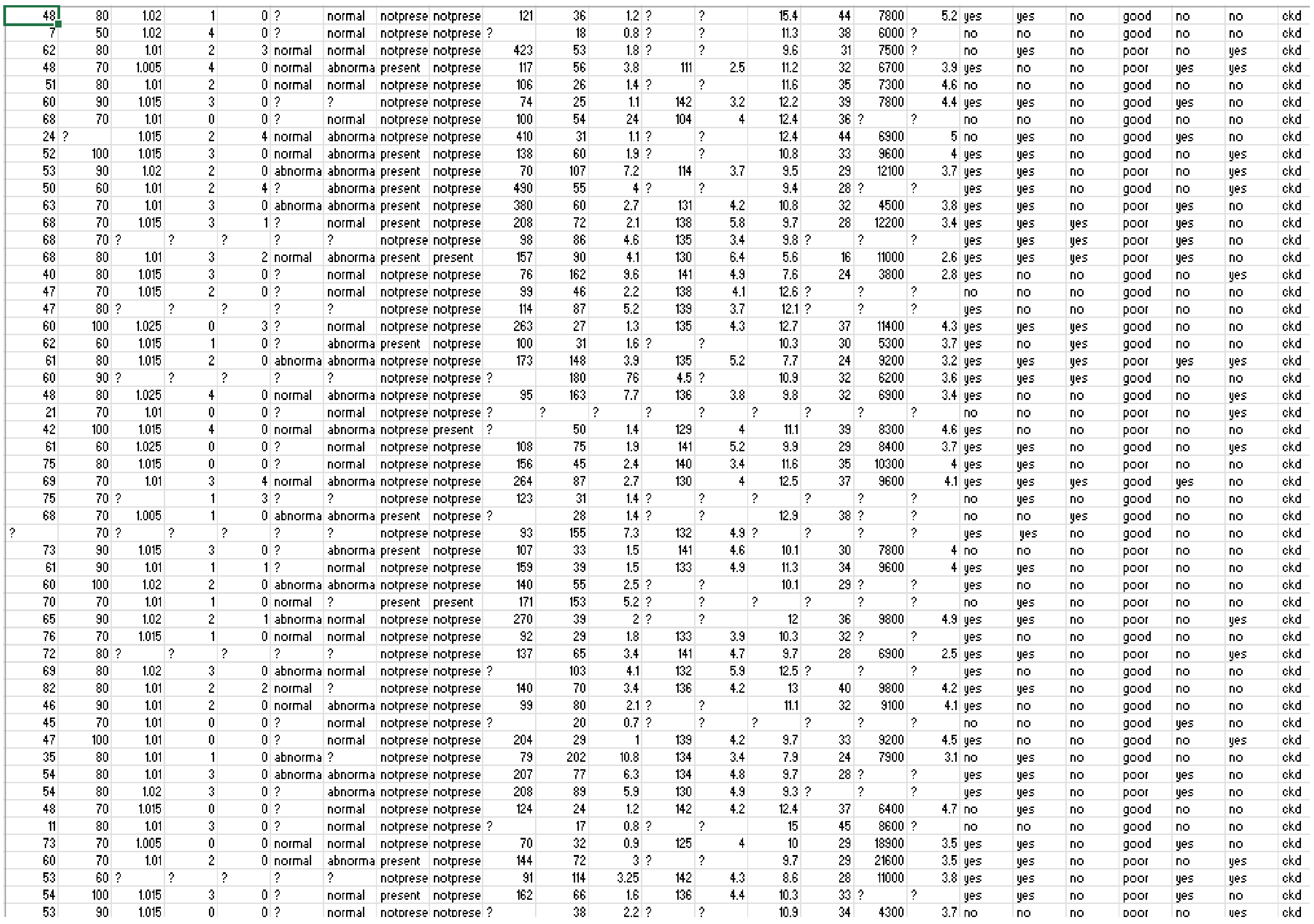

```
SVC() 0.9875
SVC(degree=2, kernel='poly') 0.85
SVC(kernel='poly') 0.75
GaussianNB() 0.9375
LogisticRegression() 0.9875
DecisionTreeClassifier() 0.975
RandomForestClassifier() 0.9625
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 2ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯2/4 [==============>...............] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 26ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 1ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 2ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 5ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 2ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ 2/10 [=====>........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 1
0ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯2/4 [==============>...............] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 27ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ 2/10 [=====>........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 1
0ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 3ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 1ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 2ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 1ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯2/4 [==============>...............] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯4/4 [==============================] - 0s 27ms/step
 1/10 [==>...........................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯10/10 [==============================] - 0s 1ms/step
1/4 [======>.......................] - ETA: 0s▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
```

Fig 7.8



48	80	1.02	1	0	?	normal	notprese	notprese	121	36	1.2	?	?	15.4	44	7800	5.2	yes	yes	no	good	no	no	ckd
7	50	1.02	4	0	?	normal	notprese	notprese	?	18	0.8	?	?	11.3	38	6000	?	no	no	no	good	no	no	ckd
62	80	1.01	2	3	normal	normal	notprese	notprese	423	53	1.8	?	?	9.6	31	7500	?	no	yes	no	poor	no	yes	ckd
48	70	1.005	4	0	normal	abnorma	present	notprese	117	56	3.8	111	2.5	11.2	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
51	80	1.01	2	0	normal	normal	notprese	notprese	106	26	1.4	?	?	11.6	35	7300	4.6	no	no	no	good	no	no	ckd
60	90	1.015	3	0	?	?	notprese	notprese	74	25	1.1	142	3.2	12.2	39	7800	4.4	yes	yes	no	good	yes	no	ckd
68	70	1.01	0	0	?	normal	notprese	notprese	100	54	24	104	4	12.4	36	?	?	no	no	no	good	no	no	ckd
24	?	1.015	2	4	normal	abnorma	notprese	notprese	410	31	1.1	?	?	12.4	44	6900	5	no	yes	no	good	yes	no	ckd
52	100	1.015	3	0	normal	abnorma	present	notprese	138	60	1.9	?	?	10.8	33	9600	4	yes	yes	no	good	no	yes	ckd
53	90	1.02	2	0	abnorma	abnorma	present	notprese	70	107	7.2	114	3.7	9.5	29	12100	3.7	yes	yes	no	poor	no	yes	ckd
50	60	1.01	2	4	?	abnorma	present	notprese	490	55	4	?	?	9.4	28	?	?	yes	yes	no	good	no	yes	ckd
63	70	1.01	3	0	abnorma	abnorma	present	notprese	380	60	2.7	131	4.2	10.8	32	4500	3.8	yes	yes	no	poor	yes	no	ckd
68	70	1.015	3	1	?	normal	present	notprese	208	72	2.1	138	5.8	9.7	28	12200	3.4	yes	yes	yes	poor	yes	no	ckd
68	70	?	?	?	?	?	notprese	notprese	98	86	4.6	135	3.4	9.8	?	?	?	yes	yes	yes	poor	yes	no	ckd
68	80	1.01	3	2	normal	abnorma	present	present	157	90	4.1	130	6.4	5.6	16	11000	2.6	yes	yes	yes	poor	yes	no	ckd
40	80	1.015	3	0	?	normal	notprese	notprese	76	162	9.6	141	4.9	7.6	24	3800	2.8	yes	no	no	good	no	yes	ckd
47	70	1.015	2	0	?	normal	notprese	notprese	99	46	2.2	138	4.1	12.6	?	?	?	no	no	no	good	no	no	ckd
47	80	?	?	?	?	?	notprese	notprese	114	87	5.2	139	3.7	12.1	?	?	?	yes	no	no	poor	no	no	ckd
60	100	1.025	0	3	?	normal	notprese	notprese	263	27	1.3	135	4.3	12.7	37	11400	4.3	yes	yes	yes	good	no	no	ckd
62	60	1.015	1	0	?	abnorma	present	notprese	100	31	1.6	?	?	10.3	30	5300	3.7	yes	no	yes	good	no	no	ckd
61	80	1.015	2	0	abnorma	abnorma	notprese	notprese	173	148	3.9	135	5.2	7.7	24	9200	3.2	yes	yes	yes	poor	yes	yes	ckd
60	90	?	?	?	?	?	notprese	notprese	?	180	76	4.5	?	10.9	32	6200	3.6	yes	yes	yes	good	no	no	ckd
48	80	1.025	4	0	normal	abnorma	notprese	notprese	95	163	7.7	136	3.8	9.8	32	6900	3.4	yes	no	no	good	no	yes	ckd
21	70	1.01	0	0	?	normal	notprese	notprese	?	?	?	?	?	?	?	?	?	no	no	no	poor	no	yes	ckd
42	100	1.015	4	0	normal	abnorma	notprese	present	?	50	1.4	129	4	11.1	39	8300	4.6	yes	no	no	poor	no	no	ckd
61	60	1.025	0	0	?	normal	notprese	notprese	108	75	1.9	141	5.2	9.9	29	8400	3.7	yes	yes	no	good	no	yes	ckd
75	80	1.015	0	0	?	normal	notprese	notprese	156	45	2.4	140	3.4	11.6	35	10300	4	yes	yes	no	poor	no	no	ckd
69	70	1.01	3	4	normal	abnorma	notprese	notprese	264	87	2.7	130	4	12.5	37	9600	4.1	yes	yes	yes	good	yes	no	ckd
75	70	?	1	3	?	?	notprese	notprese	123	31	1.4	?	?	?	?	?	?	no	yes	no	good	no	no	ckd
68	70	1.005	1	0	abnorma	abnorma	present	notprese	?	28	1.4	?	?	12.9	38	?	?	no	no	yes	good	no	no	ckd
?	70	?	?	?	?	?	notprese	notprese	93	155	7.3	132	4.9	?	?	?	?	yes	yes	no	good	no	no	ckd
73	90	1.015	3	0	?	abnorma	present	notprese	107	33	1.5	141	4.6	10.1	30	7800	4	no	no	no	poor	no	no	ckd
61	90	1.01	1	1	?	normal	notprese	notprese	159	39	1.5	133	4.9	11.3	34	9600	4	yes	yes	no	poor	no	no	ckd
60	100	1.02	2	0	abnorma	abnorma	notprese	notprese	140	55	2.5	?	?	10.1	29	?	?	yes	no	no	poor	no	no	ckd
70	70	1.01	1	0	normal	?	present	present	171	153	5.2	?	?	?	?	?	?	no	yes	no	poor	no	no	ckd
65	90	1.02	2	1	abnorma	normal	notprese	notprese	270	39	2	?	?	12	36	9800	4.9	yes	yes	no	poor	no	yes	ckd
76	70	1.015	1	0	normal	normal	notprese	notprese	92	29	1.8	133	3.9	10.3	32	?	?	yes	no	no	good	no	no	ckd
72	80	?	?	?	?	?	notprese	notprese	137	65	3.4	141	4.7	9.7	28	6900	2.5	yes	yes	no	poor	no	yes	ckd
69	80	1.02	3	0	abnorma	normal	notprese	notprese	?	103	4.1	132	5.9	12.5	?	?	?	yes	no	no	good	no	no	ckd
82	80	1.01	2	2	normal	?	notprese	notprese	140	70	3.4	136	4.2	13	40	9800	4.2	yes	yes	no	good	no	no	ckd
46	90	1.01	2	0	normal	abnorma	notprese	notprese	99	80	2.1	?	?	11.1	32	9100	4.1	yes	no	no	good	no	no	ckd
45	70	1.01	0	0	?	normal	notprese	notprese	?	20	0.7	?	?	?	?	?	?	no	no	no	good	yes	no	ckd
47	100	1.01	0	0	?	normal	notprese	notprese	204	29	1	139	4.2	9.7	33	9200	4.5	yes	no	no	good	no	yes	ckd
35	80	1.01	1	0	abnorma	?	notprese	notprese	79	202	10.8	134	3.4	7.9	24	7900	3.1	no	yes	no	good	no	no	ckd
54	80	1.01	3	0	abnorma	abnorma	notprese	notprese	207	77	6.3	134	4.8	9.7	28	?	?	yes	yes	no	poor	yes	no	ckd
54	80	1.02	3	0	?	abnorma	notprese	notprese	208	89	5.9	130	4.9	9.3	?	?	?	yes	yes	no	poor	yes	no	ckd
48	70	1.015	0	0	?	normal	notprese	notprese	124	24	1.2	142	4.2	12.4	37	6400	4.7	no	yes	no	good	no	no	ckd
11	80	1.01	3	0	?	normal	notprese	notprese	?	17	0.8	?	?	15	45	8600	?	no	no	no	good	no	no	ckd
73	70	1.005	0	0	normal	normal	notprese	notprese	70	32	0.9	125	4	10	29	18900	3.5	yes	yes	no	good	yes	no	ckd
60	70	1.01	2	0	normal	abnorma	present	notprese	144	72	3	?	?	9.7	29	21600	3.5	yes	yes	no	poor	no	yes	ckd
53	60	?	?	?	?	?	notprese	notprese	91	114	3.25	142	4.3	8.6	28	11000	3.8	yes	yes	no	poor	yes	yes	ckd
54	100	1.015	3	0	?	normal	present	notprese	162	66	1.6	136	4.4	10.3	33	?	?	yes	yes	no	poor	yes	no	ckd
53	90	1.015	0	0	?	normal	notprese	notprese	?	38	2.2	?	?	10.9	34	4300	3.7	no	no	no	poor	no	yes	ckd

Fig 7.9

8.Testing

8.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

8.2 Types Of Testing

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive.. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination.

Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	:interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It has a purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, specification or requirements document. It is a test in which the software under test is treated as a black box .you cannot “see” into it.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Example of test cases

TC ID	Condition Being Tested	Expected Result	Result
UITC_CkOt1	path of the dataset file	Display path	pass
UITC_CkOt2	if null values found replace with new values during preprocessing	Display sum of null values for each attribute and remove them or replace them	pass
UITC_CkOt3	Display the extracted features from the dataset	Display features and their count	Passed
UITC_CkOt4	Display the accuracy score	Accuracy score and evolution metrics in results	passed

8.3 Test Cases:

- Logistic Regression

  Training Results:0.977%

  Testing Results:0.976%

  Accuracy Logistic Regression:0.9875
- Decision Tree Classifiers

  Training Results:1.000%

  Testing Results:0.945%

  Accuracy Decision Tree Classifiers:0.975
- Random Forest Classifiers

  Training Results:1.000%

  Testing Results:0.973%

  Accuracy Random Forest Classifiers:0.9625
- Support Vector Machines

  Training Results:0.976%

  Testing Results:0.978%

  Accuracy Support Vector Machines:0.9875
- Artificial Neural Networks

  Training Results:1.000%

  Testing Results:0.99%

  Accuracy Artificial Neural Networks:0.9
- Gaussian naive bayes

  Training Results:0.973%

  Testing Results:0.979%

  Accuracy Gaussian naive bayes:0.9375
- K -nearest neighbours

  Training Results:1.000%

  Testing Results:0.974%

  Accuracy K-nearest neighbours:0.85

8.4 Execution Steps:

Step-1 Data Collection:

Gather a dataset containing relevant information about patients, including clinical and demographic features, as well as information about whether they have chronic kidney disease or not.

Step-2 Data Preprocessing:

Clean the data by handling missing values, outliers, and formatting issues.

Encode categorical variables into numerical representations if necessary.

Step-3 Data Exploration:

Perform exploratory data analysis (EDA) to gain insights into the dataset.

Visualize data distributions, correlations, and other relevant statistics.

Step-4 Feature Selection/Engineering:

Choose the most relevant features for your model.

Create new features if they can provide additional predictive power.

Step-5 Split Data:

Split the dataset into training and testing sets. A common split is 80% for training and 20% for testing.

Step-6 Model Selection:

Choose an appropriate machine learning algorithm for binary classification (since you're predicting the presence or absence of kidney disease). Common choices include Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, and more.

Step-7 Model Training:

Train the selected model using the training dataset.

Step-8 Model Evaluation:

Evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.

Step-9 Hyperparameter Tuning:

Fine-tune the model's hyperparameters to optimize its performance. You can use techniques like grid search or random search.

Step-10 Cross-Validation:

Perform k-fold cross-validation to ensure the model's generalization performance.

Step-11 Model Interpretation :

If needed, interpret the model's decisions to understand what features are influencing the predictions.

Step-12 Model Deployment :

If you intend to deploy the model, you can save it to a file for later use.

9. Conclusion

In conclusion, you can develop a predictive model for chronic kidney disease using machine learning in Python by following the steps mentioned above. This approach does not involve creating a web application but focuses on building a standalone model for prediction. The success of your model will depend on the quality of your dataset, the choice of the machine learning algorithm, and your ability to fine-tune hyperparameters. It's essential to thoroughly evaluate your model's performance and make sure it is providing accurate predictions before applying it in a real-world medical setting. Additionally, staying up-to-date with the latest research and medical guidelines is crucial to ensure that your model's predictions align with current medical knowledge.

10. Future Scope

Personalized Medicine: Future models can focus on personalizing treatment plans based on a patient's CKD risk profile, genetic information, and lifestyle factors.

Real-time Monitoring: Develop systems for real-time monitoring of CKD patients, which can provide early warnings and alerts based on IoT and wearable device data.

Integration with Electronic Health Records (EHR): Integrating machine learning models with EHR systems can provide clinicians with decision support tools and better insights into patient histories.

Telemedicine and Mobile Apps: Develop mobile applications that allow patients to monitor their CKD risk and communicate with healthcare providers for remote consultations.

Genomic and Proteomic Data Integration: Incorporate genomic and proteomic data to understand the genetic predisposition and biomarkers associated with CKD.

Data Sharing and Collaboration: Collaboration and data sharing among healthcare institutions and researchers can lead to the development of more robust CKD prediction models.

Explainable AI: Future models should focus on providing more interpretable and transparent results to gain trust from medical professionals and patients.

Early Detection and Prevention: The development of more accurate and robust models can lead to early detection of CKD, which is crucial for effective prevention and intervention.

Future scope of CKD prediction using ML using Python:

Develop more accurate ML models: As ML algorithms continue to improve, it is likely that ML models will become even more accurate at predicting CKD.

Develop ML models that can predict CKD earlier: Currently, ML models are best at predicting CKD in patients who are already showing signs of the disease. However, it would be even more beneficial to develop ML models that can predict CKD earlier, before any symptoms develop.

Develop ML models that can be used to personalize treatment plans: ML models can be used to analyze patient data and identify factors that are associated with a higher risk of CKD complications. This information can then be used to develop personalized treatment plans that can help to reduce the risk of complications.

Develop ML models that can be used to improve public health: ML models can be used to analyze large datasets of population data to identify trends and patterns in CKD risk factors. This information can then be used to develop public health interventions that can help to reduce the prevalence of CKD.

Overall, the future of CKD prediction using ML using Python is very bright. ML has the potential to revolutionize the way that CKD is diagnosed and treated.

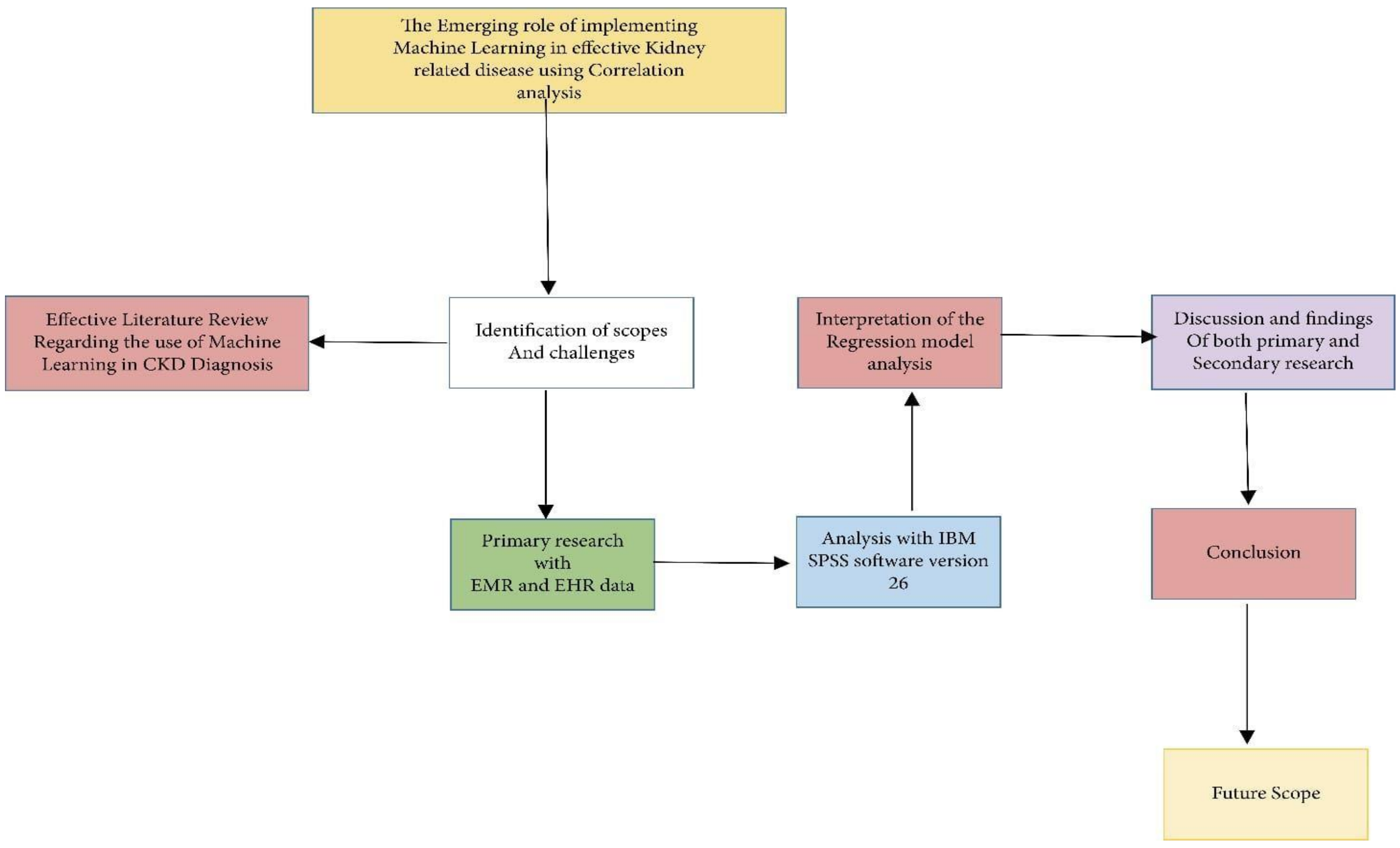

The Emerging role of implementing Machine Learning in effective Kidney related disease using Correlation analysis
Effective Literature Review Regarding the use of Machine Learning in CKD Diagnosis
Identification of scopes And challenges
Interpretation of the Regression model analysis
Discussion and findings Of both primary and Secondary research
Primary research with EMR and EHR data
Analysis with IBM SPSS software version 26
Conclusion
Future Scope

Fig 10.1

11. References

[1] Chaurasia, V.,Pal, S., Tiwari,B. (2018). Chronic Kidney Disease: A Predictive model using Decision Tree.

[2] Nimmala, S., Ramadevi, Y., Sahith, R., & Cheruku, R. (2018).High blood pressure prediction based on AAA++ using machine learning algorithms.

[3] Golino, H. F., Amaral, L. S. D. B., Duarte, S. F. P., Gomes, C. M. A., Soares, T. D. J., Reis, L. A. D., & Santos, J. (2014). Predicting Increased Blood Pressure Using Machine Learning. Journal of Obesity, 2014, 1–12.

[4] of the World Kidney Day Steering Committ (2017) Obesity and kidney disease: hidden consequences of the epidemic

[5] Kerr M., Bray B., Medcalf J., Matthews B., (2012), “Estimating the financial cost of chronic kidney disease to the NHS in England”, Nephrology Dialysis Transplantation, 27(3)

[6] Practical Points for Use of Estimated GFR and Albuminuria (ACR) in Assessing CKD (2015),Guidelines and Audit Implementation Network

[7] Vijayarani D.S., Dhayanand M.S. , (2015), “Kidney Disease Prediction Using SVM and ANN Algorithms”, International Journal of Computing and Business Research (IJCBR), 6(2)

[8] Sheikh Salahuddin Ahmed, Md. Aminul Haque Khan, Tarafdar Runa Laila .(2013).Treatment and Prevention of Common Complications of Chronic Kidney Disease

[9] Teng S. , Du H. , Wu N. , Zhang W. , Su J. , (2010), “A Cooperative Network Intrusion Detection Based on Fuzzy SVMs”, Journal of Networks, 5(4), 475-483

[10] Moguerza J. M. , Javier M. ,Muñoz A. , (2006), "Support Vector Machines with Applications", Stat. Sci, 21(3), 322-336

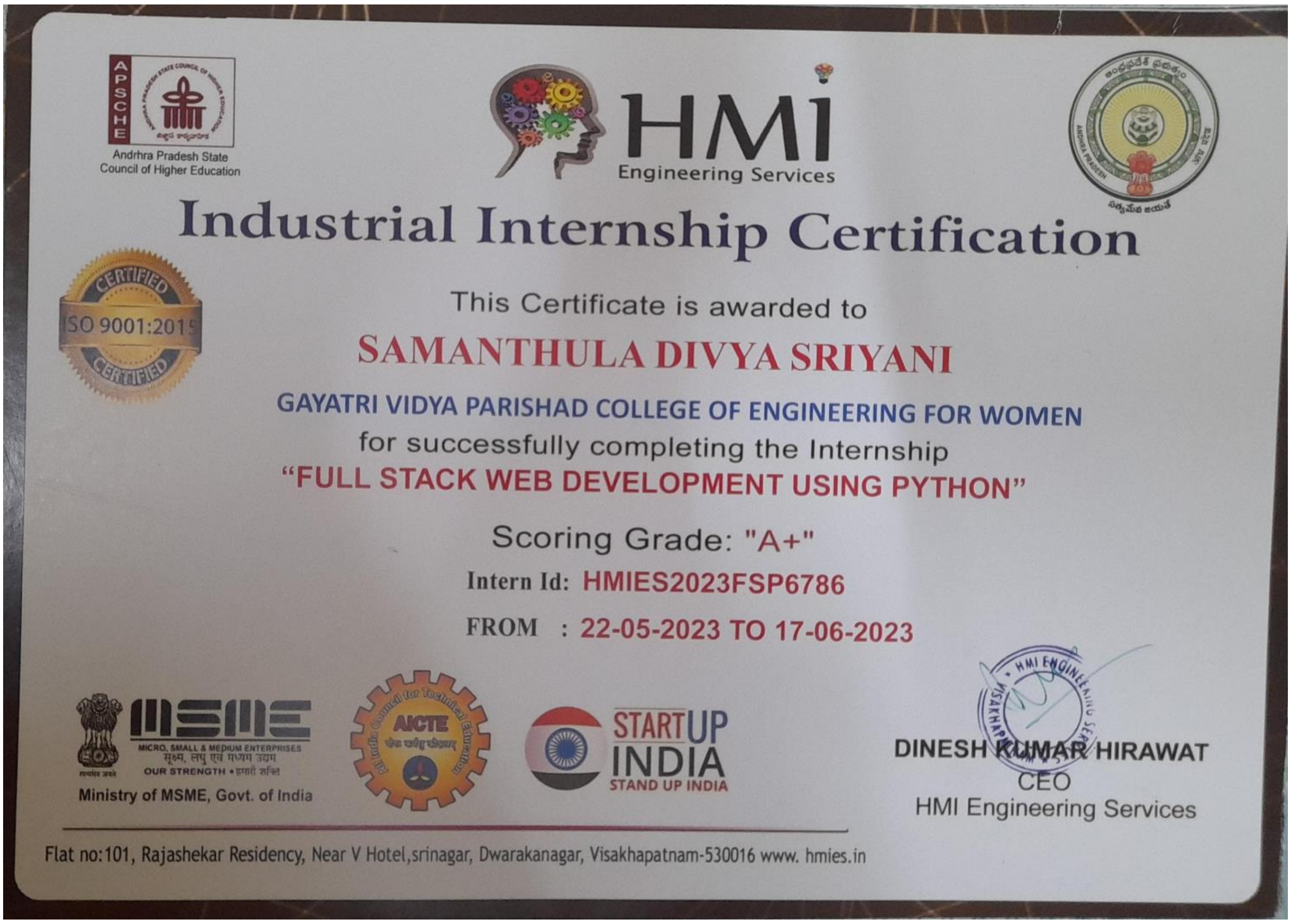
[11] Patel S., (2017), “Chapter 2 : SVM (Support Vector Machine) — Theory”, Machine Learning 101, Retrieved from https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machinetheory-f0812effc72

[12] The Renal Association, (2019), “CKD stages”, Retrieved from https://renal.org/informationresources/the-uk-eckd-guide/ckd-stages/

[13] Rakel D., (2018), ”Integrative Medicine 4thedition)”, Philadelphia, PA: Elsevier

[14] Tests to Measure Kidney Function, Damage and Detect Abnormalities, (n.d.), National Kidney Foundation, Retrieved from https://www.kidney.org/atoz/content/kidneytests

12. Certificate



APSCHE
Andrhra Pradesh State Council of Higher Education

HMI Engineering Services

Industrial Internship Certification

ISO 9001:2015 CERTIFIED

This Certificate is awarded to

SAMANTHULA DIVYA SRIYANI

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

for successfully completing the Internship

"FULL STACK WEB DEVELOPMENT USING PYTHON"

Scoring Grade: "A+"

Intern Id: HMIES2023FSP6786

FROM : 22-05-2023 TO 17-06-2023

MSME MICRO, SMALL & MEDIUM ENTERPRISES सूक्ष्म, लघु एवं मध्यम उद्यम OUR STRENGTH • हमारी शक्ति
Ministry of MSME, Govt. of India

AICTE All India Council for Technical Education

STARTUP INDIA STAND UP INDIA

HMI ENGINEERING SERVICES VISAKHAPATNAM

DINESH KUMAR HIRAWAT
CEO
HMI Engineering Services

Flat no:101, Rajashekar Residency, Near V Hotel,srinagar, Dwarakanagar, Visakhapatnam-530016 www. hmies.in