CLASSIFICATION OF CHRONIC KIDNEY DISEASE USING DATA MINING TECHNIQUES

BY

A.FARHEEN

(16PCS005)

Project Report Submitted

In Partial fulfillment of the requirements for the award of

Master's Degree in Computer Science

Department of Computer Science,

Avinashilingam Institute for Home Science and Higher Education for Women, (Deemed to be University),

Coimbatore-641043

April-2019

CLASSIFICATION OF CHRONIC KIDNEY DISEASE USING DATA MINING TECHNIQUES

BY
A.FARHEEN

(16PCS005)

Project Report Submitted

In Partial fulfillment of the requirements for the award of

Master's Degree in Computer Science

Department of Computer Science,

Avinashilingam Institute for Home Science and Higher Education for Women, (Deemed to be University),

Coimbatore-641043

Signature of the Head of the Department **Signature of the Supervisor**

Viva Voce Examination Held on: _____

Signature of the Examiners

ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for his constant love and grace that he showered upon me.

I would like to express my deep sense of reverential gratitude and sincere thanks to **Padma Shri Dr. P. R. Krishnakumar, Chancellor,**Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for his support and encouragement during the course of my study.

I owe my great deal of gratitude to **Dr. (Mrs.) V.PremavathyVijayan,Vice Chancellor**, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for extending all resources that facilitated the conduct of the present work.

I express my gratitude to **Dr. (Mrs.) S. Kowsalya, Registrar**, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities necessary for the work.

I also thankful to **Dr. (Mrs.) K. Udhaya Chandrika, Dean, School of Physical Sciences and computational Sciences,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for granting the facility required.

I wish to place my deep sense of gratitude to **Dr. (Mrs.)V. Radha, Professor and Head, Department of Computer Science,** for providing all the facilities required to complete the project.

I heartily thank my esteemed project guide **Dr. (Mrs.) S.N.Geethalakshmi, Professor, Department of Computer Science**, for imparting the tremendous assistance and well-timed support for triumph of my project

I express my honorable thanks to my project coordinator **Dr. (Mrs.) G.Padmavathi, Professor, Department of Computer Science,** for her kind advice and knowledgeable suggestions which helped me to complete my project successfully.

Finally, I take pride to thank my parents and those who helped me directly or indirectly for carrying out this work.

ABSTRACT

ABSTRACT

Data Mining in Healthcare has become a present trend for obtaining accurate results of medical diagnosis; **Chronic Kidney Disease (CKD)** has become an international fitness problem and is a place of concern. It is a situation where kidneys turn out to be damaged and cannot filter toxic wastes within the frame. By using Data Mining Techniques, researchers have the scope to predict the Chronic Kidney Disease. This helps doctors to diagnose and suggest the treatment at an early stage. It also helps the patients to know about their health condition at an earlier stage and follow necessary diet and prescriptions. Healthcare Management is one of the areas which is using machine learning techniques broadly for different objectives. Chronic Kidney Disease is a growing disease in recent years and many researches are being done to predict its progression and classify the datasets based on related features. In this project, we focus on applying different machine learning classification algorithm to a dataset with 400 observations and 25 attributes for diagnosis of chronic kidney disease. The project was developed under **Python** and MS-Excel used for storing the data as CSV format.

CONTENTS

TABLE OF CONTENT

INTRODUCTION

1. INTRODUCTION

1.1 Data Mining

Data mining is the process of analyzing data and discovering useful information. Sometimes it is called knowledge Discovery. Data mining a non-trivial extraction of implicit, previously unknown and potentially useful information from data. Data mining is the process of analyzing the given data in order to provide useful information contained in that data that cannot be retrieved through queries. . It is an application that view data from different angles and group it into information that is useful in many perspectives.

Data mining involves six common classes of tasks:

- **Anomaly detection (Outlier/change/deviation detection)** – The identification of unusual data records, that might be interesting or data errors that require further investigation.
- **Association rule learning (Dependency modelling)** – Searches for relationships between variables. For example, a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.
- **Clustering** – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.
- **Classification** – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".
- **Regression** – attempts to find a function which models the data with the least error.
- **Summarization** – providing a more compact representation of the data set, including visualization and report generation.

1.2 Medical Data Mining:

Data Mining is particularly useful in medical field when no availability of evidence favoring a particular treatment option is found. Large amount of complex data is being generated by healthcare industry about patients, diseases, hospitals, medical equipments, claims, treatment cost etc. that requires processing and analysis for knowledge extraction.Data mining comes up with a set of tools and techniques which when applied to this processed data, provides knowledge to healthcare professionals for making appropriate decisions and enhancing the performance of patient management tasks. Patients with similar health issues can be grouped together and effective treatment plans could be suggested based on patient's history, physical examination, diagnosis and previous treatment patterns.

Chronic kidney disease (CKD) has become a global health issue and is an area of concern. It is a condition where kidneys become damaged and cannot filter toxic wastes in the body. This project work focuses on detecting life threatening diseases like Chronic Kidney Disease (CKD) using Classification algorithms .

1.3 Objective of the project

The objective of the project is to classify chronic kidney disease from the affected patient's records as per the variations and severity of the disease being affected.

1.4 Overview of the Project

Classification is the method of predicting the final results will be based on the given input dataset. The set of rules attempts to discover relationships among the attributes that might make it feasible to expect the outcome. Then, the performance of the classification algorithms on the Chronic Kidney Disease dataset are compared in terms of accuracy measures.

SYSTEM SPECIFICATION

2. SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION

- Hard Disk : 285 GB
- Monitor : 21’ Color with VGI card support
- RAM : 4.00 GB
- Processor : Intel(R) Core(TM) i3
- Processor speed : CPU M 350 @ 2.27GHz
- System Type : 64-bit Operating System

2.2 SOFTWARE SPECIFICATION

Operating System : Windows 10

Software : Python 3.7(Anaconda3)

2.3 ABOUT THE SOFTWARE

2.3.1 Python Programming

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** : Python is processed at runtime by the interpreter. Do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** : one can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** :Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** : Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

- Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

PYTHON FEATURES

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – one can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

PYTHON LIBRARIES

- Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.
- Scrapy. If one are involved in webscraping then this is a must have library. After using this library won't use any other.
- wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. One will really love it.
- Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

- SQLAlchemy. A database library. Many love it and many hate it.
- BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.
- Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.
- NumPy. How can we leave this very important library. It provides some advance math functionalities to python.
- SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
- matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
- Pygame. Which developer does not like to play games and develop them. This library will help one achieve your goal of 2d game development.
- Pyglet. A 3d animation and game creation engine. This is the engine in which the famous
- python port of minecraft was made
- pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.
- pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.
- Scapy. A packet sniffer and analyzer for python made in python.
- pywin32. A python library which provides some useful methods and classes for interacting with windows.
- nltk. Natural Language Toolkit – realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But it's capacity is beyond that.
- nose. A testing framework for python. It is used by millions of python developers. It is a must have if one do test driven development.
- SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

- IPython. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that one take a look at it.
- Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.
- Scrapy. If one are involved in webscraping then this is a must have library. After using this library one won't use any other.
- wxPython. A gui toolkit for python. primarily used it in place of tkinter.
- Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
- BeautifulSoup. xml and html parsing library is very useful for beginners.
- Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.
- NumPy. It provides some advance math functionalities to python.
- SciPy. When talk about NumPy then have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
- matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
- Pygame. Which developer does not like to play games and develop them ? This library will help to achieve your goal of 2d game development.
- Pyglet. A 3d animation and game creation engine.
- pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.
- pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.
- Scapy. A packet sniffer and analyzer for python made in python.
- pywin32. A python library which provides some useful methods and classes for interacting with windows.
- nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

- SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.
- IPython. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more.

USES OF PYTHON

Python are widely used for

- System programming
- GUI Programming
- Internet scripting
- Database Programming
- Gaming,Robotics
- Mozilla
- Government and non-profit organization

2.3.2 TABLEAU SOFTWARE

Tableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data into the very easily understandable format.

Data analysis is very fast with Tableau and the visualizations created are in the form of dashboards and worksheets. The data that is created using Tableau can be understood by professional at any level in an organization. It even allows a non-technical user to create a customized dashboard.

The best feature Tableau are

- Data Blending
- Real time analysis
- Collaboration of data

The great thing about Tableau software is that it doesn't require any technical or any kind of programming skills to operate. The tool has garnered interest among the people from all sectors such as business, researchers, different industries, etc.

Tableau Product Suite

The Tableau Product Suite consists of

- Tableau Desktop
- Tableau Public
- Tableau Online
- Tableau Server
- Tableau Reader

Data analytics in tableau can be classified into two section

1. **Developer Tools:** The Tableau tools that are used for development such as the creation of dashboards, charts, report generation, visualization fall into this category. The Tableau products, under this category, are the Tableau Desktop and the Tableau Public.
2. **Sharing Tools**: As the name suggests, the purpose of the tool is sharing the visualizations, reports, dashboards that were created using the developer tools. Products that fall into this category are Tableau Online, Server, and Reader.

2.4 About the Dataset:

The clinical data of 400 records considered for analysis has been taken from **UCI Machine Learning Repository**. The data obtained after cleaning and removing missing values is 220. There are 25 attributes in the dataset. The numerical attributes include age, blood pressure, blood glucose random, blood urea, serum creatinine, sodium, potassium, hemoglobin, packaged cell volume, WBC count, RBC count. The nominal attributes include specific gravity, albumin, sugar, RBC, pus cell, pus cell clumps, bacteria, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema, anemia and class.

Data set Description :

age - age

bp - blood pressure

sg - specific gravity

al - albumin

su - sugar

rbc - red blood cells

pc - pus cell

pcc - pus cell clumps

ba - bacteria

bgr - blood glucose random

bu - blood urea

sc - serum creatinine

sod - sodium

pot - potassium

hemo - hemoglobin

pcv - packed cell volume

wc - white blood cell count

rc - red blood cell count

htn - hypertension

dm - diabetes mellitus

cad - coronary artery disease

appet - appetite

pe - pedal edema

ane - anemia

class – class

Attribute information :

We use 24 + class = 25 (11 numeric ,14 nominal)

1. Age(numerical) age in years

2. Blood Pressure(numerical) bp in mm/Hg

3. Specific Gravity(nominal) sg - (1.005,1.010,1.015,1.020,1.025)

4. Albumin(nominal) al - (0,1,2,3,4,5)

5. Sugar(nominal) su - (0,1,2,3,4,5)

6. Red Blood Cells(nominal) rbc - (normal,abnormal)

7. Pus Cell (nominal) pc - (normal,abnormal)

8. Pus Cell clumps(nominal) pcc - (present,notpresent)

9. Bacteria(nominal) ba - (present,notpresent)

10. Blood Glucose Random(numerical) bgr in mgs/dl

11. Blood Urea(numerical) bu in mgs/dl

12. Serum Creatinine(numerical) sc in mgs/dl

13. Sodium(numerical) sod in mEq/L

14. Potassium(numerical) pot in mEq/L

15. Hemoglobin(numerical) hemo in gms

16. Packed Cell Volume(numerical)

17. White Blood Cell Count(numerical) wc in cells/cumm

18. Red Blood Cell Count(numerical) rc in millions/cmm

19. Hypertension(nominal) htn - (yes,no)

20. Diabetes Mellitus(nominal) dm - (yes,no)

21. Coronary Artery Disease(nominal) cad - (yes,no)

22. Appetite(nominal) appet - (good,poor)

23. Pedal Edema(nominal) pe - (yes,no)

24. Anemia(nominal) ane - (yes,no)

25. Class (nominal) class - (ckd,notckd)

METHODOLOGY

3. METHODOLOGY

3.1 Proposed system

The project **"Classification of Chronic Kidney Disease using Data Mining Techniques"** describes the proposed methodology for data mining from **CKD** dataset. The very first and important step is **preprocessing** and cleaning the data. Cleaning process is the process of filling the missing values based on either categorical or nominal data. In second step, pre-processed data is classified using support vector machine and decision tree algorithm features for both original and normalized data. Finally, in performance measures compare the results of accuracy to classify which is the best algorithm.

3.2 Load the Dataset

Downloading chronic kidney disease (ckd) dataset in UCI repository and loading the dataset. The chronic kidney disease (ckd) dataset is collected from the UCI repository and it contains 400 instances and 25 attributes.

3.3 Preprocessing

Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.

Data pre-processing activities includes a lot of steps. The following are some of the task involved data cleaning, integration, transformation, reduction and discretization. Data cleaning involve smooth noisy data and resolving inconsistencies where data integration includes using multiple files. Data transformation is an important aspect which includes normalization and aggregation. Data reduction and discretization are similar only that reduction includes reducing the volume but producing the same or similar analytical results whereas discretization is replacing numerical attributes with nominal ones.

3.4 Classification Algorithms

Classification techniques are applied on all features and selected features for both original and normalized data. Data is split into training set and testing set. 70 % of the data is used for training the model and rest 30 % is used for testing. Different classification algorithms such Decision Tree and Support Vector Machine are applied on the original data and normalized data.

3.4.1 Support Vector Machine (SVM)

In machine learning, Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

3.4.2 Decision Tree

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data (training data).

3.5 Performance Measures

The Confusion matrix is one of the most intuitive and easiest (unless of course, you are not confused) metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes.

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made. Precision is the ratio of the correctly classified cases to the total number of misclassified cases and correctly classified cases. Recall is the ratio of correctly classified cases to the total number of unclassified cases and correctly classified cases. F1-score might be a better measure to use if we need to seek a balance between precision and recall and there is an uneven class distribution.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

EXPERIMENTAL RESULTS

4. EXPERIMENTAL RESULTS

The experiment is performed in Python environment. Python is a multi-paradigm functional, imperative, object-oriented programming language.

4.1 Evaluation Measures

In the experiments, 5 common different measures were used for the evaluation of the classification quality: accuracy, precision, recall, f1-score and support.

The performance evaluation of Support Vector Machine classification is listed below:

PERFORMANCE MEASURE	**CLASS : Normal**	**CLASS: Abnormal**
Accuracy	0.911764	0.911764
Precision	0.8900	1.000
Recall	1.0000	0.7000
F1-score	0.9400	0.8200
Support	24	10

The performance evaluation of Decision Tree classification is listed below:

PERFORMANCE MEASURE	**CLASS : Normal**	**CLASS: Abnormal**
Accuracy	0.961538	0.961538
Precision	0.9700	1.0000
Recall	1.0000	0.9300
F1-score	0.9900	0.9700
Support	36	15

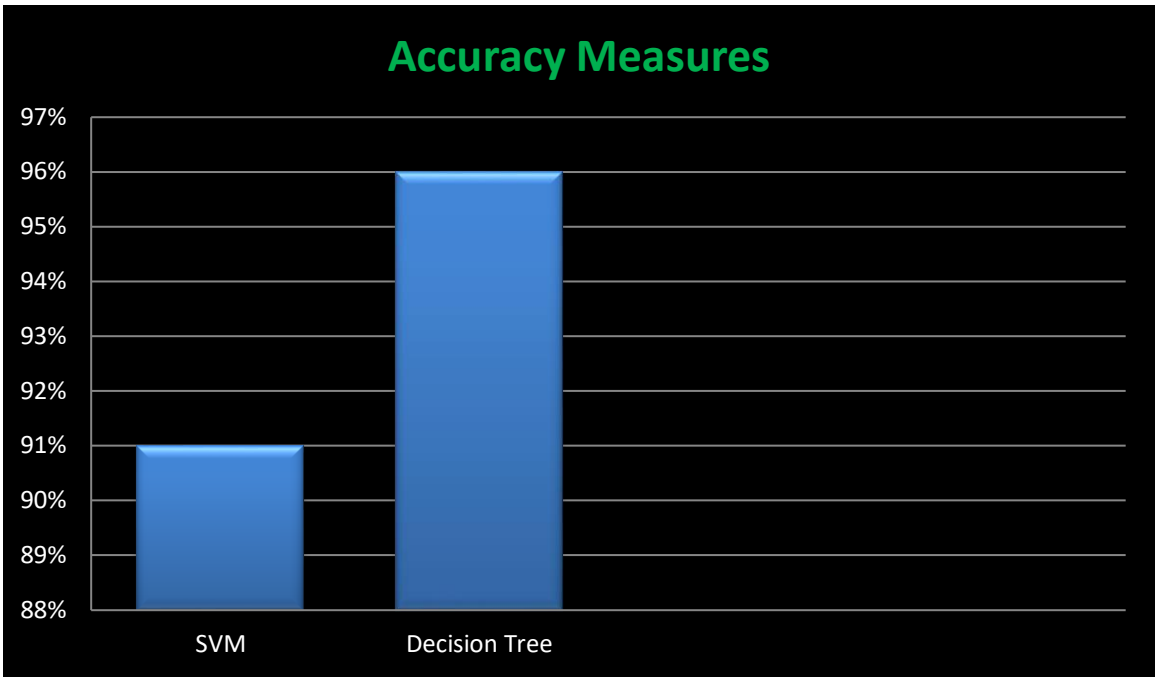


Figure 1: Accuracy measures for Classification Algorithm

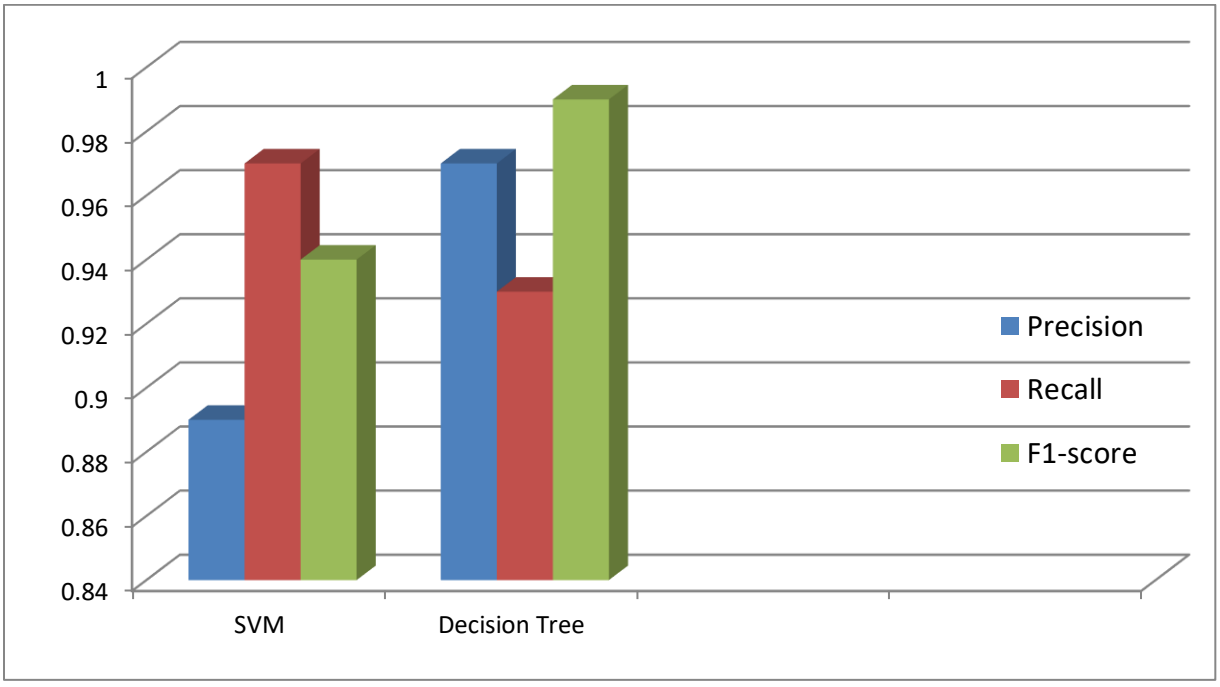


Figure 2: Performance Measures for Classification Algorithm

CONCLUSION

5. CONCLUSION

The project is based on healthcare which has been developed by using data mining techniques. For classification of Chronic Kidney Disease (CKD) Data mining techniques has been used. Classification algorithm which is used for prediction of the disease. Chronic Kidney Disease has been predicted and diagnosed using data mining classifiers such as Support Vector Machine and Decision Tree. Performances of these algorithms are compared using accuracy measures. Accuracy was calculated for each data mining algorithm. Support vector machine (SVM) got the accuracy result of **91.11%** and Decision tree algorithm got the accuracy result of **96.15%**.Comparison is made for the classification techniques; it indicates that the degree of accuracy of the Decision tree is more than Support vector machine (SVM) algorithm. Hence **Decision Tree** is more accurate for classification of Chronic Kidney Disease. This technique allows the doctors to suggest the medicine. The main aim of the project was to classify appropriately.

SCOPE FOR FUTURE ENHANCEMENT

6. SCOPE FOR FUTURE ENHANCEMENT

In future work we can implement different clustering and classification algorithm for the same application and different healthcare dataset and calculating the accuracy between different algorithms and find out which algorithm is more efficient.

BIBLIOGRAPHY

7. BIBLIOGRAPHY

REFERENCE BOOKS:

- Ms. AsthaAmeta,Ms. Kalpana Jain, "Data Mining Techniques for the Prediction of Kidney Diseases and Treatment: A Review", International Journal Of Engineering And Computer Science, Feb. 2017
- J Chitra Devi, "Binary Decision Tree Classification based onC4.5 and KNN Algorithm for Banking Application", International Journal of Computational Intelligence and Informatics
- Agarwal, Y., & Pandey, H. M. (2014, September). Performance evaluation of different techniques in the context of data mining-A case of an eye disease. In*Confluence The Next Generation Information Technology Summit (Confluence), 2016 5th International Conference*-(pp. 72-76). IEEE.
- Ranganatha, S., Raj, H. P., Anusha, C., & Vinay, S. K. (2013). Medical data mining and analysis for kidney disease dataset using classificationtechniques.
- Tahmasebian S, Ghazisaeedi M, Langarizadeh M, Mokhtaran M, Mahdavi-Mazdeh M, Javadian P. Applying data mining techniques to determine important parameters in chronic kidney disease and the relations of these parameters to each other. J Renal Inj Prev. 2017;6(2):83-87. DOI: 10.15171/jrip.2017.16.

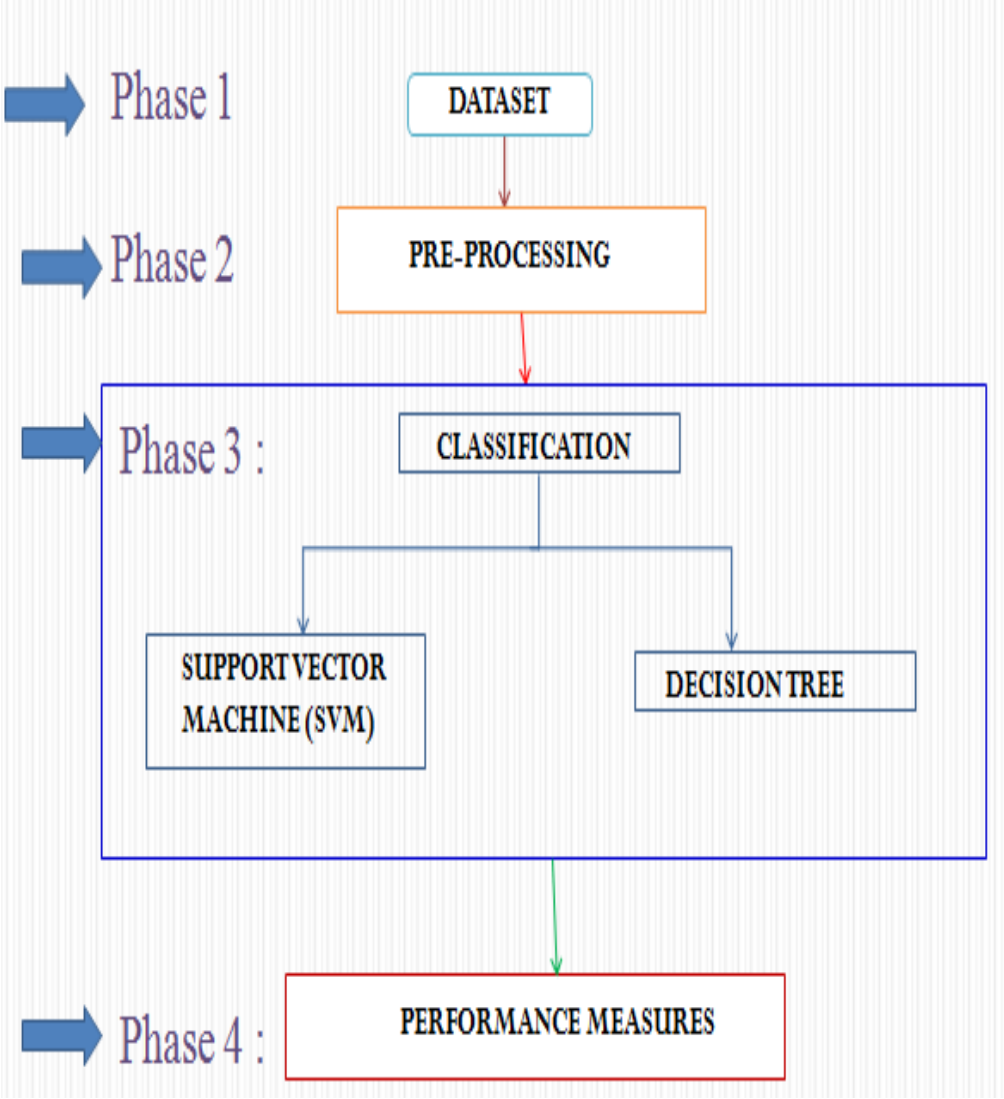
REFERENCE SITES:

- https://machinelearningmastery.com
- https://www.researchgate.net/publication/328028191_Analysis_and_Prediction_of_Chronic_Kidney_Disease_using_Data_Mining_Techniques
- https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease
- https://acadgild.com/blog/decision-tree-python-code
- https://pythonprogramming.net/support-vector-machine-intro-machine-learning-tutorial/

APPENDIX

8. APPENDIX

8.1DATA FLOW DIAGRAM



8.2 SCREEN SHOTS

Load dataset

```
In [2]: ckd_data = pd.read_csv('kidney_disease_orig.csv')
        ckd_data.shape

Out[2]: (400, 26)
```

	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe
2	1	0		normal	notpresen	notpresen	121	36	1.2			15.4	44	7800	5.2	yes	yes	no	good	no
3	4	0		normal	notpresen	notpresent		18	0.8			11.3	38	6000		no	no	no	good	no
4	2	3	normal	normal	notpresen	notpresen	423	53	1.8			9.6	31	7500		no	yes	no	poor	no
5	4	0	normal	abnormal	present	notpresen	117	56	3.8	111	2.5	11.2	32	6700	3.9	yes	no	no	poor	yes
6	2	0	normal	normal	notpresen	notpresen	106	26	1.4			11.6	35	7300	4.6	no	no	no	good	no
7	3	0			notpresen	notpresen	74	25	1.1	142	3.2	12.2	39	7800	4.4	yes	yes	no	good	yes
8	0	0		normal	notpresen	notpresen	100	54	24	104	4	12.4	36			no	no	no	good	no
9	2	4	normal	abnormal	notpresen	notpresen	410	31	1.1			12.4	44	6900	5	no	yes	no	good	yes
10	3	0	normal	abnormal	present	notpresen	138	60	1.9			10.8	33	9600	4	yes	yes	no	good	no
11	2	0	abnormal	abnormal	present	notpresen	70	107	7.2	114	3.7	9.5	29	12100	3.7	yes	yes	no	poor	no
12	2	4		abnormal	present	notpresen	490	55	4			9.4	28			yes	yes	no	good	no
13	3	0	abnormal	abnormal	present	notpresen	380	60	2.7	131	4.2	10.8	32	4500	3.8	yes	yes	no	poor	yes
14	3	1		normal	present	notpresen	208	72	2.1	138	5.8	9.7	28	12200	3.4	yes	yes	yes	poor	yes
15					notpresen	notpresen	98	86	4.6	135	3.4	9.8				yes	yes	yes	poor	yes
16	3	2	normal	abnormal	present	present	157	90	4.1	130	6.4	5.6	16	11000	2.6	yes	yes	yes	poor	yes
17	3	0		normal	notpresen	notpresen	76	162	9.6	141	4.9	7.6	24	3800	2.8	yes	no	no	good	no
18	2	0		normal	notpresen	notpresen	99	46	2.2	138	4.1	12.6				no	no	no	good	no
19					notpresen	notpresen	114	87	5.2	139	3.7	12.1				yes	no	no	poor	no
20	0	3		normal	notpresen	notpresen	263	27	1.3	135	4.3	12.7	37	11400	4.3	yes	yes	yes	good	no
21	1	0		abnormal	present	notpresen	100	31	1.6			10.3	30	5300	3.7	yes	no	yes	good	no
22	2	0	abnormal	abnormal	notpresen	notpresen	173	148	3.9	135	5.2	7.7	24	9200	3.2	yes	yes	yes	poor	yes
23					notpresen	notpresent		180	76	4.5		10.9	32	6200	3.6	yes	yes	yes	good	no

Figure 1: Load the Dataset

```
In [5]: #Drop sample with NaN data(i.e with no values)
        ckd_data = ckd_data.dropna(axis=0)
        ckd_data['class'].value_counts()
        ckd_data.shape

Out[5]: (169, 25)

In [6]: X = ckd_data.drop('class', axis=1)
        y = ckd_data['class']

In [7]: ckd_data.head()

Out[7]:
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	class
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	32	6700	3.9	1.0	0.0	0.0	0	1	1	1.0
9	53.0	90.0	1.020	2.0	0.0	1.0	1.0	1.0	0.0	70.0	...	29	12100	3.7	1.0	1.0	0.0	0	0	1	1.0
11	63.0	70.0	1.010	3.0	0.0	1.0	1.0	1.0	0.0	380.0	...	32	4500	3.8	1.0	1.0	0.0	0	1	0	1.0
14	68.0	80.0	1.010	3.0	2.0	0.0	1.0	1.0	1.0	157.0	...	16	11000	2.6	1.0	1.0	1.0	0	1	0	1.0
20	61.0	80.0	1.015	2.0	0.0	1.0	1.0	0.0	0.0	173.0	...	24	9200	3.2	1.0	1.0	1.0	0	1	1	1.0

5 rows × 25 columns

Figure 2: Preprocessing the Dataset

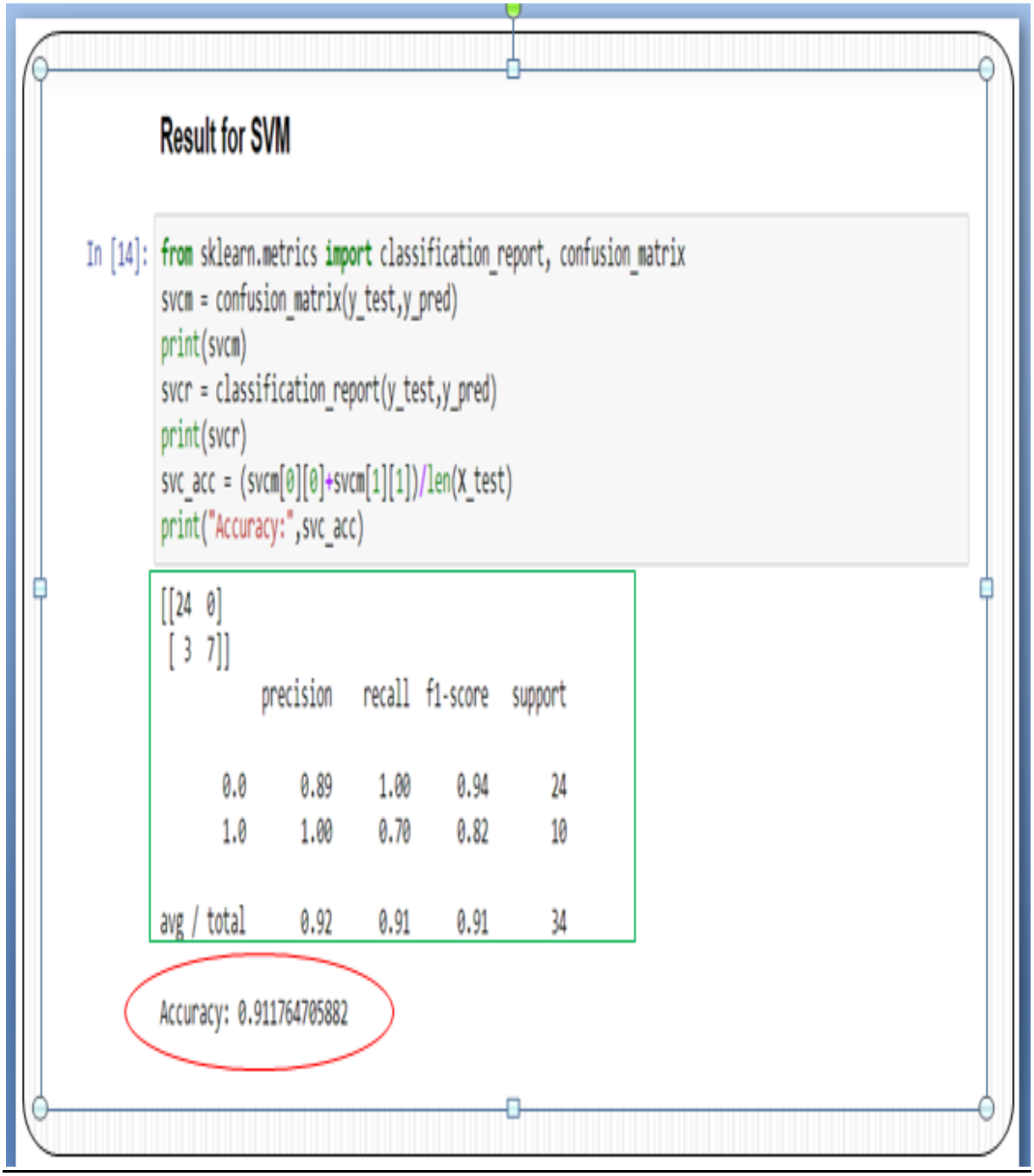


Figure 3: Result of support Vector Machine (SVM) Algorithm

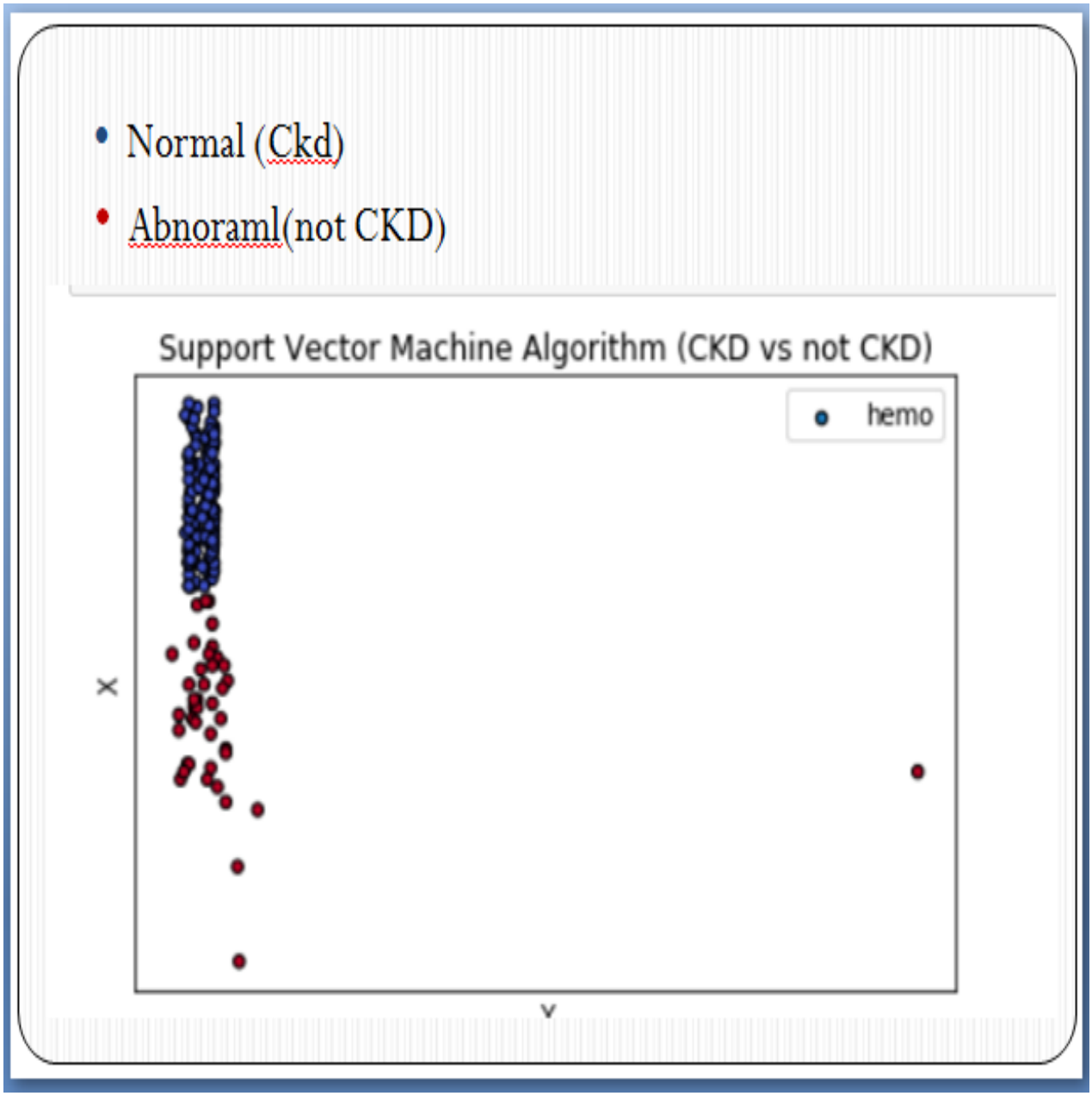


Figure 4: Visualization Plot for Support Vector Machine (SVM)

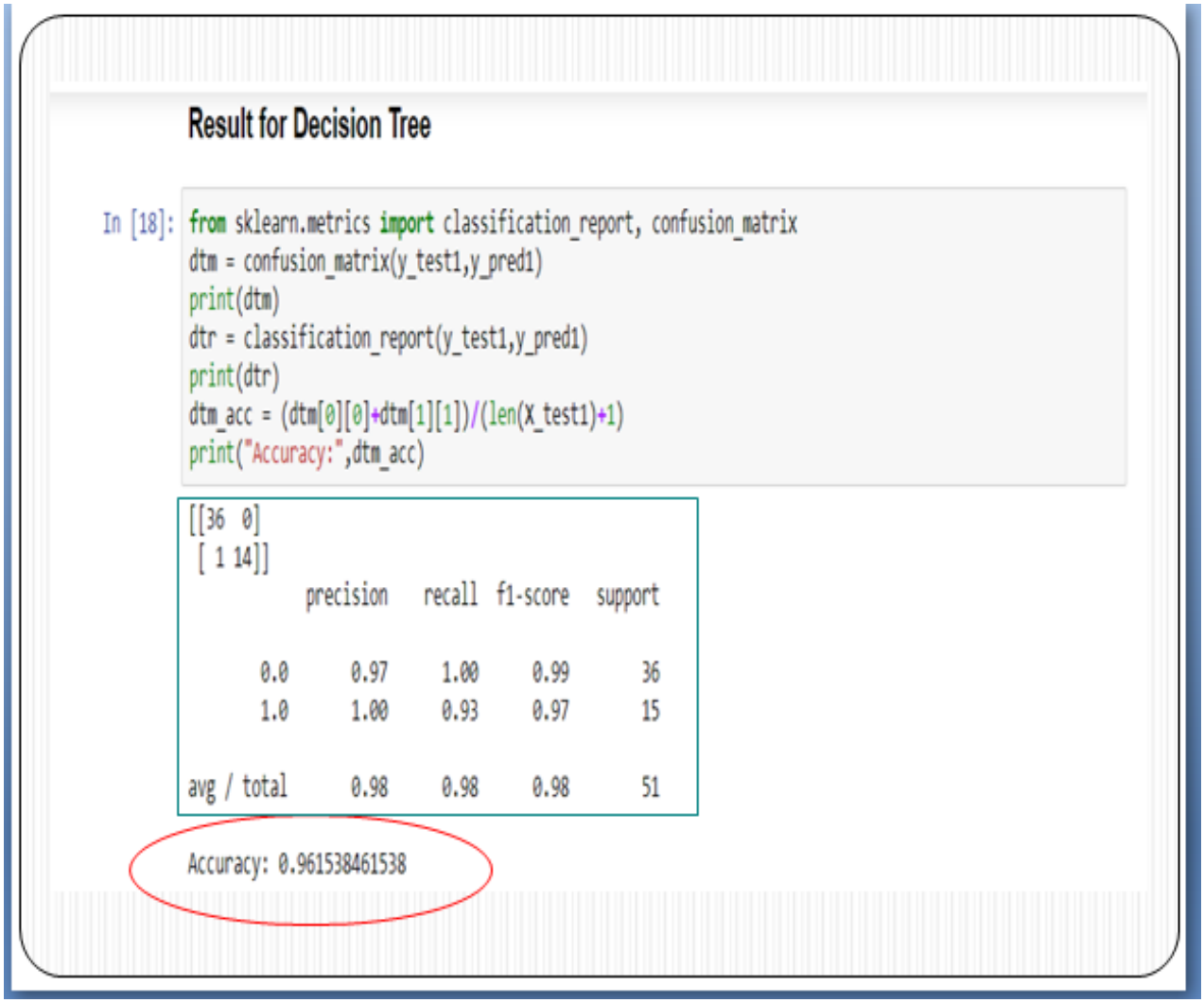


Figure 5: Result for Decision Tree Algorithm

Figure 6: Performance Measures of Classification Algorithm

Pseudo Code

```
#Load the Dataset

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.metrics import roc_curve, auc, confusion_matrix,
classification_report,accuracy_score

%matplotlib inline

from sklearn.datasets.samples_generator import make_blobs

ckd_data = pd.read_csv('kidney_disease_orig.csv')

ckd_data.shape


 #Preprocessing

ckd_data[['htn','dm','cad','pe','ane']] =
ckd_data[['htn','dm','cad','pe','ane']].replace(to_replace={'yes':1,'no':0})

ckd_data[['rbc','pc']] = ckd_data[['rbc','pc']].replace(to_replace={'abnormal':1,'normal':0})

ckd_data[['pcc','ba']] =
ckd_data[['pcc','ba']].replace(to_replace={'present':1,'notpresent':0})
```

```
ckd_data[['appet']] =
ckd_data[['appet']].replace(to_replace={'good':1,'poor':0,'no':np.nan})

ckd_data['classification'] =
ckd_data['classification'].replace(to_replace={'ckd':1.0,'ckd\t':1.0,'notckd':0.0,'no':0.0})

ckd_data.rename(columns={'classification':'class'},inplace=True)

ckd_data['pe'] = ckd_data['pe'].replace(to_replace='good',value=0) # Not having pedal
edema is good

ckd_data['appet'] = ckd_data['appet'].replace(to_replace='no',value=0)

ckd_data['cad'] = ckd_data['cad'].replace(to_replace='\tno',value=0)

ckd_data['dm'] = ckd_data['dm'].replace(to_replace={'\tno':0,'\tyes':1,' yes':1, '':np.nan})

ckd_data.drop('id',axis=1,inplace=True)

ckd_data = ckd_data.dropna(axis=0)

ckd_data['class'].value_counts()

ckd_data.shape

X = ckd_data.drop('class', axis=1)

y = ckd_data['class']

ckd_data.head()


# Split the Dataset

from sklearn.model_selection import train_test_split

X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size = 0.20)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

```
# Training Support Vector Machine(SVM)

# Call SVM using scikit-learn and train the model on the train data

from sklearn.svm import SVC

svclassifier = SVC(kernel = 'linear',C = 0.5,gamma = 0.001)

svclassifier.fit(X_train, y_train)


# Prediction Using SVM

# do prediction on the test data

y_pred = svclassifier.predict(X_test)

print(y_pred)


#Result for SVM

from sklearn.metrics import classification_report, confusion_matrix

svcm = confusion_matrix(y_test,y_pred)

print(svcm)

svcr = classification_report(y_test,y_pred)

print(svcr)

svc_acc = (svcm[0][0]+svcm[1][1])/len(X_test)

print("Accuracy:",svc_acc)
```

```
#Data Visualization for SVM
def make_meshgrid(x, y, h=.02):
  x_min, x_max = x.min() - 1, x.max() + 1
   y_min, y_max = y.min() - 1, y.max() + 1
  xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
  return xx, yy
def plot_contours(ax, clf, xx, yy, **params):
 Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
 out = ax.contourf(xx, yy, Z, **params)
 return out
# Plotting only 2 features for visualization
          fig, ax = plt.subplots()
          # Set-up grid for plotting.
          X0, X1 = X.iloc[:, 13], X.iloc[: ,14]
          xx, yy = make_meshgrid(X0, X1)
#plot_contours(ax, svclassifier, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
          ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
          ax.set_ylabel('X')
```

```
        ax.set_xlabel('Y')
        ax.set_xticks(())
        ax.set_yticks(())
        ax.set_title('Support Vector Machine Algorithm (CKD vs not CKD)')
        ax.legend()
        plt.show()


#Training Decision Tree
# Call Decision Tree using scikit-learn and train the model on the train data
from sklearn import tree
dtclassifier = tree.DecisionTreeClassifier()
dtclassifier.fit(X_train, y_train)


#Prediction Using Decision Tree
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size = 0.20)
# do prediction on the test data
y_pred1 = dtclassifier.predict(X_test1)
print(y_pred1)
```

```
#Result for Decision Tree

from sklearn.metrics import classification_report, confusion_matrix

dtm = confusion_matrix(y_test1,y_pred1)

print(dtm)

dtr = classification_report(y_test1,y_pred1)

print(dtr)

dtm_acc = (dtm[0][0]+dtm[1][1])/(len(X_test1)+1)

print("Accuracy:",dtm_acc)



#Performance Comparison

if(dtm_acc>svc_acc):

    print("Decision Tree's accuracy is better")

elif(svc_acc>dtm_acc):

    print("Support Vector Machine's accuracy is better")

else:

    print("Both Support Vector Machine and Decision Tree performs equally")
```