

URINARY BIOMARKERS FOR PANCREATIC CANCER PREDICTION USING DATA SCIENCE TECHNIQUE

A PROJECT REPORT

Submitted by

VISHWA S [REGISTER NO: 211419104310]

THARUN C [REGISTER NO: 211419104291]

SANDEEP V [REGISTER NO: 211419104230]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report “**URINARY BIOMARKERS FOR PANCREATIC CANCER PREDICTION USING DATA SCIENCE TECHNIQUE**” is the bonafide work of “**VISHWA S (211419104310), THARUN C (211419104291), SANDEEP V (211419104230)**” who carried out the project work under my supervision.

SIGNATURE

**Dr.L.JABASHEELA M.E.,Ph.D.
HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

**MRS. VIJAYALAKSHMI.P,
M.C.A.,M.Phil.,M.Tech.,
ASSISTANT PROFESSOR
GRADE 1**

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) were examined in the End Semester

Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We **VISHWA S (211419104310), THARUN C (211419104291), SANDEEP V (211419104230)** hereby declare that this project report titled “ **URINARY BIOMARKERS FOR PANCREATIC CANCER PREDICTION USING DATA SCIENCE TECHNIQUE**”, under the guidance of **MRS.VIJAYALAKSHMI.P ,M.C.A.,M.Phil.,M.Tech** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

VISHWA S

THARUN C

SANDEEP V

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.**, for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr.L.JABASHEELA,M.E.,Ph.D.**, For the support extended throughout the project.

We would like to thank our project coordinator **Mr.M.MOHAN, M.Tech.,(Ph.d)** and our project guide **Mrs.VIJAYALAKSHMI.P,M.C.A, M.Phil.,M.Tech** and all the faculty members of the department of CSE for their advice and encouragement for the successful completion of the project

VISHWA S

THARUN C

SANDEEP V

ABSTRACT

Pancreatic cancer is the fourth most common cancer-related cause of death in the United States and is usually asymptomatic in early stages. There is a scarcity of tests that facilitate early diagnosis or accurately predict the disease progression. To this end, biomarkers have been identified as important tools in the diagnosis and management of pancreatic cancer. Despite the increasing number of biomarkers described in the literature, most of them have demonstrated moderate sensitivity and specificity and are far from being considered as screening tests. More efficient non-invasive biomarkers are needed to facilitate early-stage diagnosis and interventions. Multi-disciplinary collaboration might be required to facilitate the identification of such markers. Data mining is a commonly used technique for processing enormous data. Researchers apply several data mining and machine learning techniques to analyse huge complex data, to helping the prediction of pancreatic cancer. Different algorithms are compared and the best model is used for predicting the outcome.

TABLE OF CONTENTS

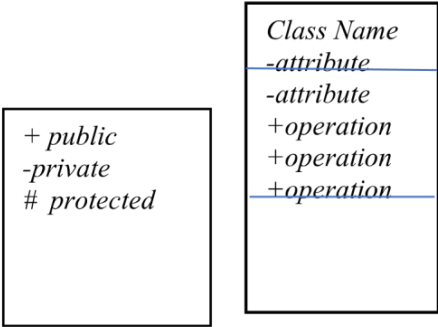
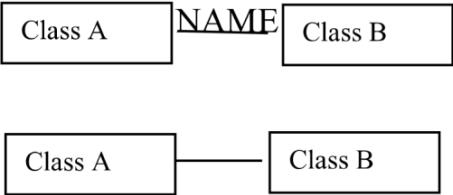
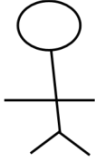
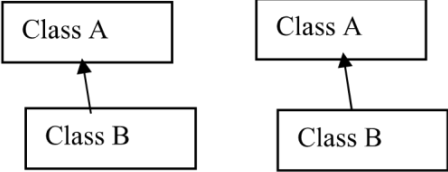
CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF SYMBOLS	ix
1.	INTRODUCTION	2
	1.1 Domain Overview	2
	1.1.1.Data Science	
	1.1.2.Artificial Intelligence	
	1.1.3.Machine Learning	
	1.2 Problem Statement	7
2.	LITERATURE SURVEY	9
3.	SYSTEM ANALYSIS	14
	3.1 Existing System	14
	3.1.1 Disadvantages	
	3.2 Proposed system	14
	3.2.1 Advantages	
	3.3 Feasibility Study	15
	3.4 Requirement Specification	17
	3.4.1 Software Requirements	
	3.4.2 Hardware Requirements	
	3.5 Software Description	18
	3.5.1 Anaconda Navigator	
	3.5.2 Jupyter Notebook	
	3.5.3 Python	

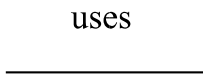
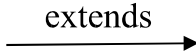

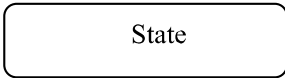
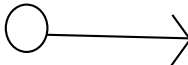
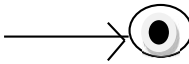
4.	SYSTEM DESIGN	29
	4.1 Work flow Diagram	29
	4.2 Use Case Diagram	30
	4.3 Class Diagram	31
	4.4 Activity Diagram	32
	4.5 Sequence Diagram	33
	4.6 Entity Relationship Diagram (ERD)	34
5.	SYSTEM ARCHITECTURE	36
	5.1 Modules	36
	5.1.1 Module Description	
	5.2 Algorithm and Techniques	44
	5.3 DEPLOYMENT	50
6.	SYSTEM IMPLEMENTATION	58
	6.1 Coding	58
7.	SYSTEM TESTING	73
	7.1 Unit Testing	73
	7.2 Integration Testing	73
8.	CONCLUSION	76
	APPENDIX	78
	Screenshot	
	REFERENCES	80

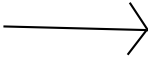
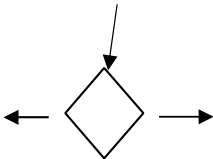
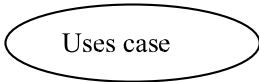
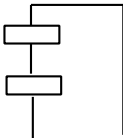
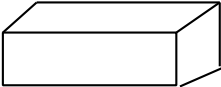
LIST OF FIGURES

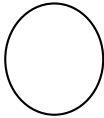


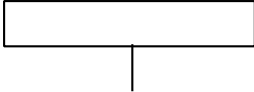
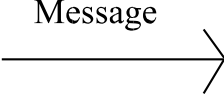
FIGURE NO.	FIGURE NAME	PAGE NO.
Figure 1.1	PROCESS OF MACHINE LEARNING	6
Figure 3.1	ARCHITECTURE OF PROPOSED MODEL	15
Figure 3.2	PROCESS OF DATA FLOW DIAGRAM	17
Figure 3.3	ANACONDA NAVIGATOR	20
Figure 3.4	ANACONDA NAVIGATOR	20
Figure 4.1	WORK FLOW DIAGRAM	29
Figure 4.2	USE CASE DIAGRAM	30
Figure 4.3	CLASS DIAGRAM	31
Figure 4.4	ACTIVITY DIAGRAM	32
Figure 4.5	SEQUENCE DIAGRAM	33
Figure 4.6	ENTITY RELATIONSHIP DIAGRAM	34
Figure 5.1	SYSTEM ARCHITECTURE	36
Figure 5.2	DATA PRE-PROCESSING MODULE	39
Figure 5.3	DATA VIRTUALIZATION MODULE	40
Figure 5.4	K-NEAREST NEIGHBOUR MODULE	46
Figure 5.5	RANDOM FOREST CLASSIFIER MODULE	47
Figure 5.6	AdaBoost MODULE DIAGRAM	48
Figure 5.7	DECISION TREE MODULE	49
Figure 9.1	SAMPLE OUTPUT SCREEN	78
Figure 9.2	SAMPLE OUTPUT SCREEN	78
Figure 9.3	SAMPLE OUTPUT SCREEN	78

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external

			environment
5.	Relation(uses)		Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object

11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interaction between the system and external environment.
14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components

16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
			Represents external entities such as
17.	External entity		keyboard, sensors etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Pancreatic cancer is one of the most lethal malignant tumors, characterized by delayed diagnosis, difficult treatment, and high mortality. The overall five year survival rate among patients is less than 9%. In the process of diagnosis and treatment, an accurate segmentation of pancreatic cancer plays an important role. Attiyeh et al. performed 3D modeling based on accurately segmented pancreatic cancer, calculated the tumor size, extracted texture features, and built survival prediction models for patients. Magnetic resonance-guided radiation therapy has potential advantages in treating locally advanced pancreatic cancer. The critical step in this method is contouring, whereby a target volume is generated at the beginning of each treatment stage

1.1 DOMAIN OVERVIEW

1.1.1 Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

1.1.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, and speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered on particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are

called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of common sense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

1.1.3 MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig 1.1 Process of Machine Learning

Supervised Machine Learning **is the** majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output **is $y = f(X)$** . The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression, multi-class classification, Decision Trees** and **support vector machines etc**. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

1.2 PROBLEM STATEMENT

Pancreatic cancer is the fourth most common cancer-related cause of death in the United States and is usually asymptomatic in early stages. There is a scarcity of tests that facilitate early diagnosis or accurately predict the disease progression. To this end, biomarkers have been identified as important tools in the diagnosis and management of pancreatic cancer. Despite the increasing number of biomarkers described in the literature, most of them have demonstrated moderate sensitivity and specificity and are far from being considered as screening tests. More efficient non-invasive biomarkers are needed to facilitate early-stage diagnosis and interventions. Multi-disciplinary collaboration might be required to facilitate the identification of such markers.

The goal is to develop a machine learning model for Pancreatic cancer , to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them

Review of Literature Survey

Title : Predicted Prognosis of Pancreatic Cancer Patients by Machine Learning

Author : Julius M. Kernbach, and Victor E. Staartjes

Year : 2020

We recently read the article by Yokoyama and colleagues (1), in which the authors report a predictive model integrating DNA methylation status of three mucin genes to predict overall survival at a designated 5-year interval in pancreatic cancer. They collected samples from 191 patients and compared support vector machines (SVM) with various kernels of ranging complexity and a neural network. Models were trained using k-fold cross-validation (with various choices of k) or leave-one-out cross-validation or a 50/50 split to evaluate generalizability. However, common discriminative performance measures, including AUC, F1-Score, sensitivity or specificity are not reported for neither the training nor testing set; resampling was not used consistently; model selection and evaluation was not performed separately;

model calibration was not assessed; class imbalance and imputation were not considered; approaches to combat overfitting, such as dropout in neural networks, were not included—in total, the prognostic value of the model cannot safely be evaluated on the basis of the presented results.

Title : Management of Metastatic Pancreatic Adenocarcinoma

Author: Ahmad R. Cheema, MDa,b, Eileen M. O'Reilly, MDc,d,

Year : 2016

In the year 2016, there will be approximately 53,000 estimated new individuals diagnosed with PDAC in the United States, representing approximately 3% of all cancer cases.¹ Despite the low incidence, pancreatic cancer is the fourth leading cause of cancer-related death among both men and women in the United States, with a high mortality-to-incidence ratio, and is expected to become the second leading cause of cancer-related mortality by 2030.^{2,3} PDAC is the most common histologic subtype, found in more than 85% of cases. Surgical resection is curative in a minority of patients; however, 70% to 80% of patients have unresectable disease, with more than 50% having distant metastases at the time of initial diagnosis, where the treatment goals are to control disease, palliate symptoms, and prolong survival.² For PDAC, the expected 5-year survival for all patients is approximately 6% to 7% and less for patients who are diagnosed with metastatic disease de novo.^{2,4} Putative explanations for poor outcomes are generally attributed to the asymptomatic early stages of the disease, lack of effective screening tools, early metastatic dissemination, relative resistance of the tumor to cytotoxic and targeted therapies, dense stroma, hypoxic microenvironment, and immune suppression.

Title : Diagnostic, Predictive and Prognostic Molecular Biomarkers in Pancreatic Cancer: An Overview for Clinicians

Author: Dimitrios Giannis , Dimitrios Moris , and Andrew S. Barbas

Year : 2021

Pancreatic ductal adenocarcinoma (PDAC) is the most common pancreatic malignancy and is associated with aggressive tumor behavior and poor prognosis. Most patients with PDAC present with an advanced disease stage and treatment-resistant tumors. The lack of noninvasive tests for PDAC diagnosis and survival prediction mandates the identification of

novel biomarkers. The early identification of high-risk patients and patients with PDAC is of utmost importance. In addition, the identification of molecules that are associated with tumor biology, aggressiveness, and metastatic potential is crucial to predict survival and to provide patients with personalized treatment regimens. In this review, we summarize the current literature and focus on newer biomarkers, which are continuously added to the armamentarium of PDAC screening, predictive tools, and prognostic tools.

Title : A Survey On Prediction Of Survival Time On Pancreatic Cancer Using Machine Learning Paradigms Towards Big Data

Author: Santosh Reddy P

Year : 2020

Resistance to impending disease can be created using paradigms built through previous procedures of different types, included measurable multi-variate relapses and AI. In any case, such a strategy does not offer the most predictive displays for every cases. To represent mechanized meta-training approaches that define how to predict the best performed system for all patients. The extremely selected procedure is used to maintain patient resistance. We evaluated the proposed approaches in a database of review records of careful resections of pancreatic disease.

Title : Pancreatic cancer: Clinical presentation, pitfalls and early clues

Author: E. P. DiMagno

Year : 1999

The diagnosis of pancreatic cancer usually depends upon symptoms; consequently it is late when there is no chance for cure. At this point, pain, anorexia, early satiety, sleep problems and weight loss are present. Back pain also may be prominent, which predicts unresectability and shortened survival after resection. However, earlier recognition of symptoms of pancreatic cancer might improve early detection of the cancer. For example, 25% of patients have symptoms compatible with upper abdominal disease up to 6 months prior to diagnosis and 15% of patients may seek medical attention more than 6 months prior to diagnosis. These symptoms erroneously may be attributed to problems such as irritable syndrome. Symptoms,

however, may be less common. For example a quarter of patients with pancreatic cancer may have no pain at diagnosis, and half, particularly those with pancreatic head tumors, may have little pain compared with patients with body-tail tumors. However, if the tumor is suspected because of predisposing conditions, earlier diagnosis may be possible. These conditions include diseases such as chronic pancreatitis, intraductal papillary mucinous tumor (BPMT), and recent onset of diabetes mellitus, particularly if the diabetes occurs during or beyond the sixth decade. In addition inherited syndromes also are associated with an increased risk of pancreatic cancer including familial pancreatic cancer, hereditary pancreatitis, familial adenomatous polyposis syndrome (FAP) and familial atypical multiple mole melanoma (FAMMM) syndrome (hereditary dysplastic nevus syndrome).

CHAPTER 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Existing System

Pancreatic cancer is a lethal malignant tumor with one of the worst prognoses. Accurate segmentation of pancreatic cancer is vital in clinical diagnosis and treatment. Due to the unclear boundary and small size of cancers, it is challenging to both manually annotate and automatically segment cancers. Considering 3D information utilization and small sample sizes, we propose a model-driven deep learning method for pancreatic cancer segmentation based on spiral transformation. This study developed a combined spiral-transformation and model-driven deep learning method for pancreatic cancer segmentation. A novel spiral-transformation algorithm was developed to enable a 2D model to achieve 3D image segmentation with affordable computational resources. It also provided various 2D-transformed images to realize an effective data augmentation for alleviating the problem of insufficient samples. Besides, a transformation-weight-corrected module was designed to combine with the 2D model in a unified framework for the 2D segmentation and 3D rebuilding. The extensive experiments conducted on internal multi-parametric MRI datasets and an external CT dataset demonstrated that the proposed method exhibits promising segmentation performance with strong robustness and generality, while reaching a good balance between the performance and the computational resources.

3.1.1 Disadvantages

1. It's a complex process to detect pancreatic cancer.
2. They do not calculate the performance metrics.
3. CNN captured only images, so it is not a proper way to detect cancer.
4. They are not using any machine learning algorithms.

3.2 Proposed System

The proposed method is to build a machine learning model for classification of pancreatic cancer. The process carries from data collection where the past data related to Pancreatic Cancer are collected. Data mining is a commonly used technique for processing enormous data in the healthcare domain. The Pancreatic Cancer if found before proper treatment can save lives. Machine learning is now applied and mostly used in health care

where it reduces the manual effort and better model makes error less which leads to save the life. The data analysis is done on the dataset proper variable identification done that is both the dependent variables and independent variables are found. Then proper machine learning algorithm are applied on the dataset where the pattern of data is learnt. After applying different algorithms a better algorithm is used for the prediction of outcome.

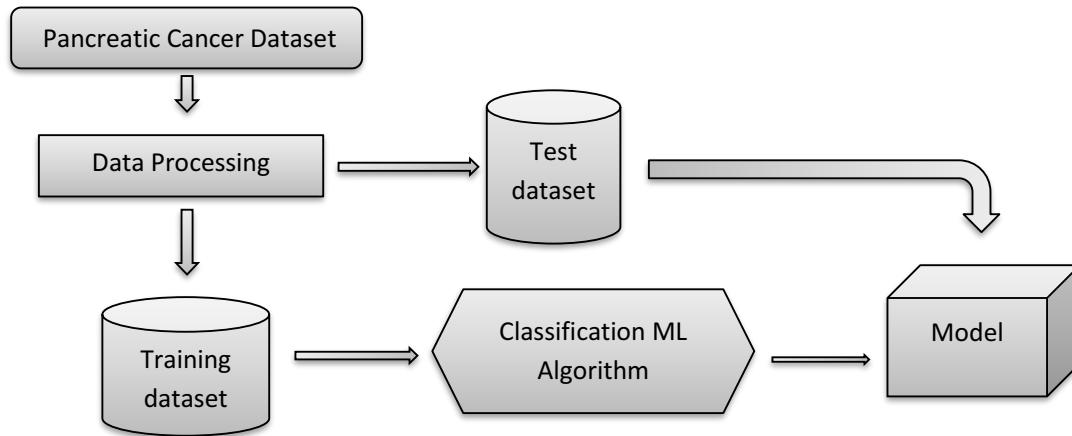


Fig 3.1 Architecture of Proposed model

3.2.1 Advantages

- The data used here will not be complex and analysis of data will be easy.
- Performance metrics of different algorithms are compared and a better prediction can be done.
- Implementing more than three algorithm for better outcome.

3.3 Feasibility study

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

Building the classification model

The prediction of pancreatic cancer, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

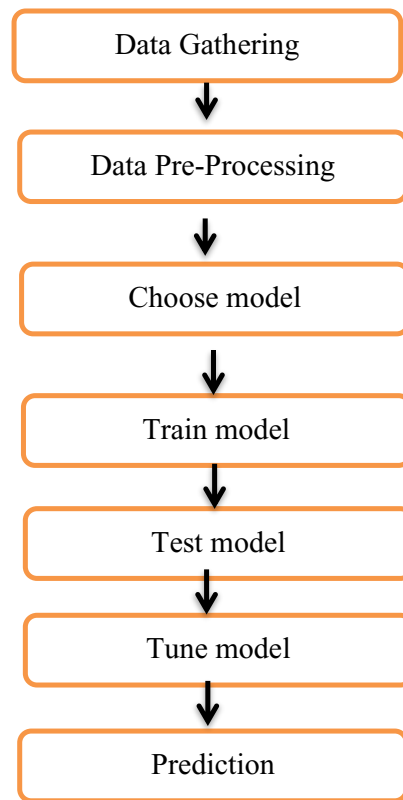


Fig 3.2 Process of dataflow diagram

3.4 REQUIREMENT SPECIFICATION

3.4.1 Software Requirements

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

3.4.2 Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

3.5 SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the `conda build` command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

3.5.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

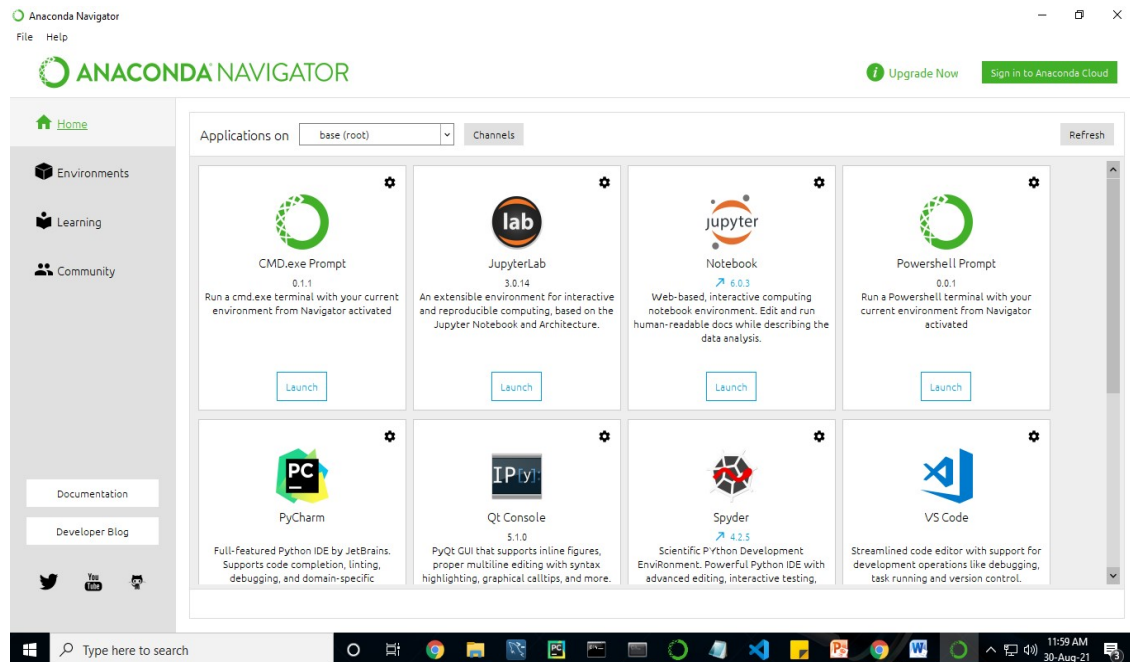


Fig 3.3

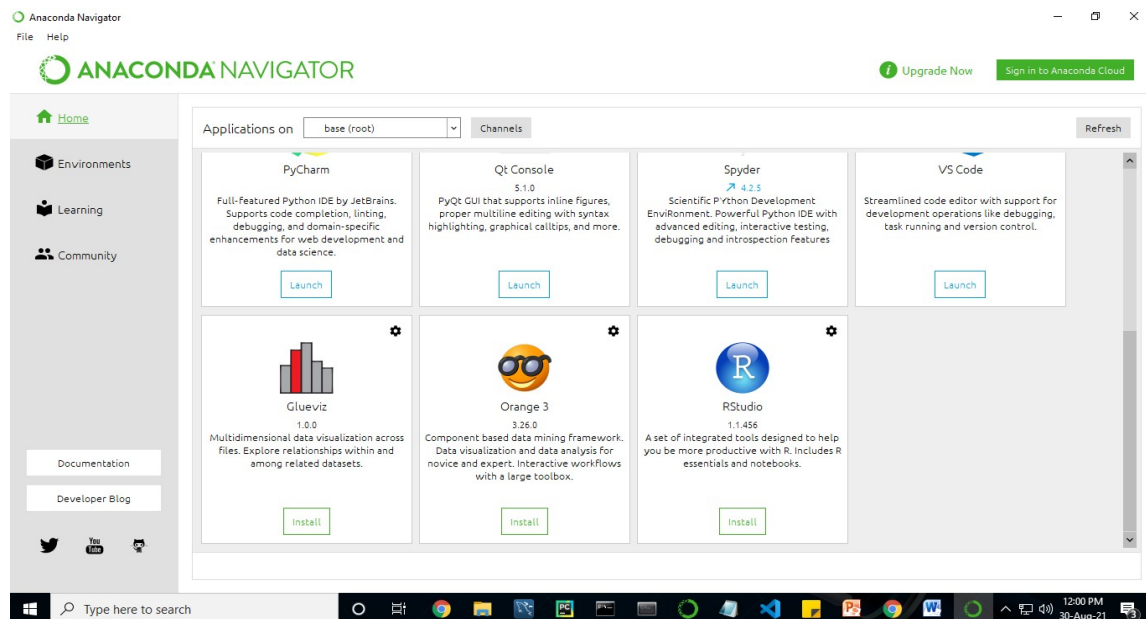


Fig 3.4

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

Conda

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. These softwares have great graphical user interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

3.5.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda

Running the Jupyter Notebook

Launching *Jupyter Notebook App*: The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- ❖ Launch the jupyter notebook app
- ❖ In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.

- ❖ Click on the menu *Help -> User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- ❖ You can run the notebook document step-by-step (one cell a time) by pressing *shift + enter*.
- ❖ You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All*.
- ❖ To restart the kernel (i.e. the computational engine), click on the menu *Kernel -> Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

Kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results.

Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

3.5.3 PYTHON

Introduction

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was

released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

History

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Work flow diagram

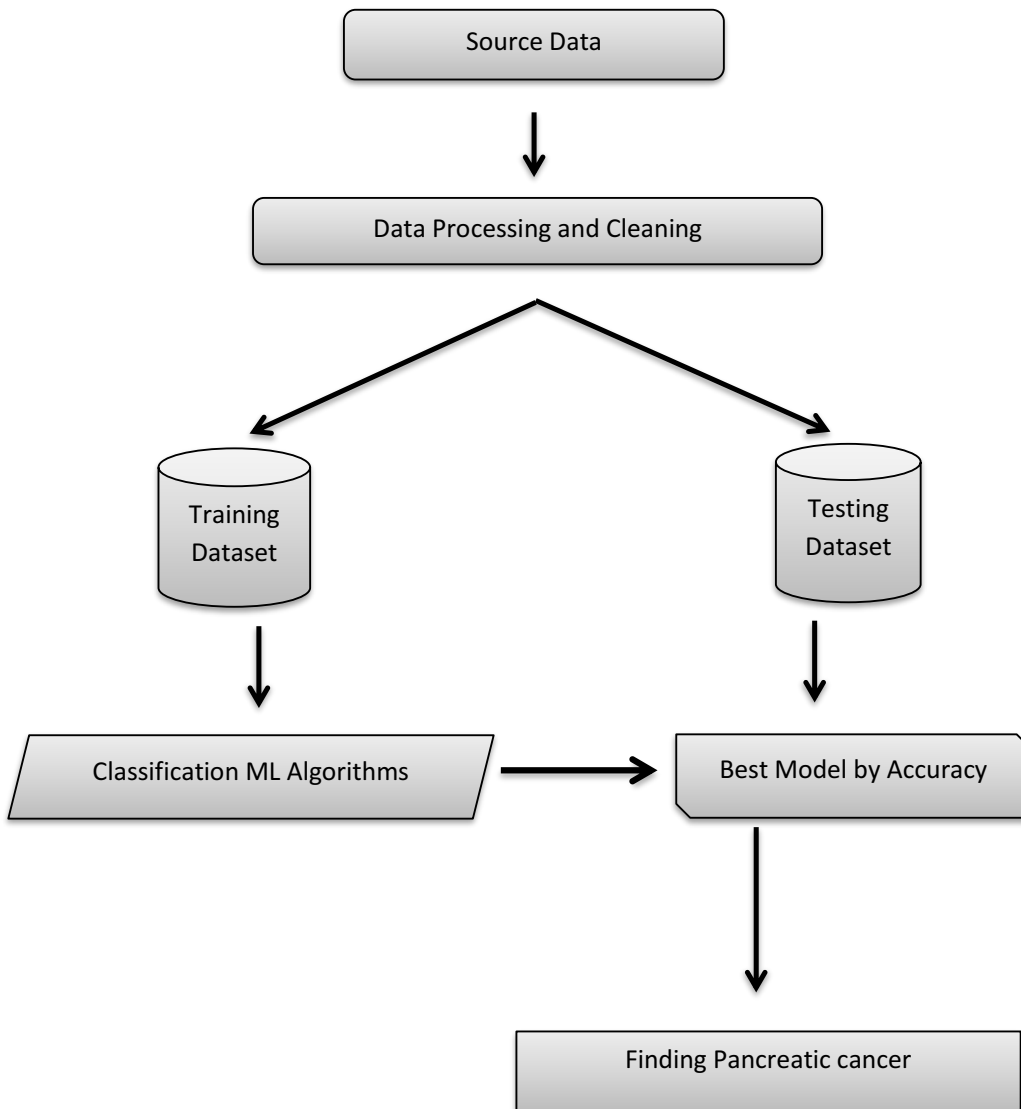


Fig 4.1 Workflow Diagram

4.2 Use Case Diagram

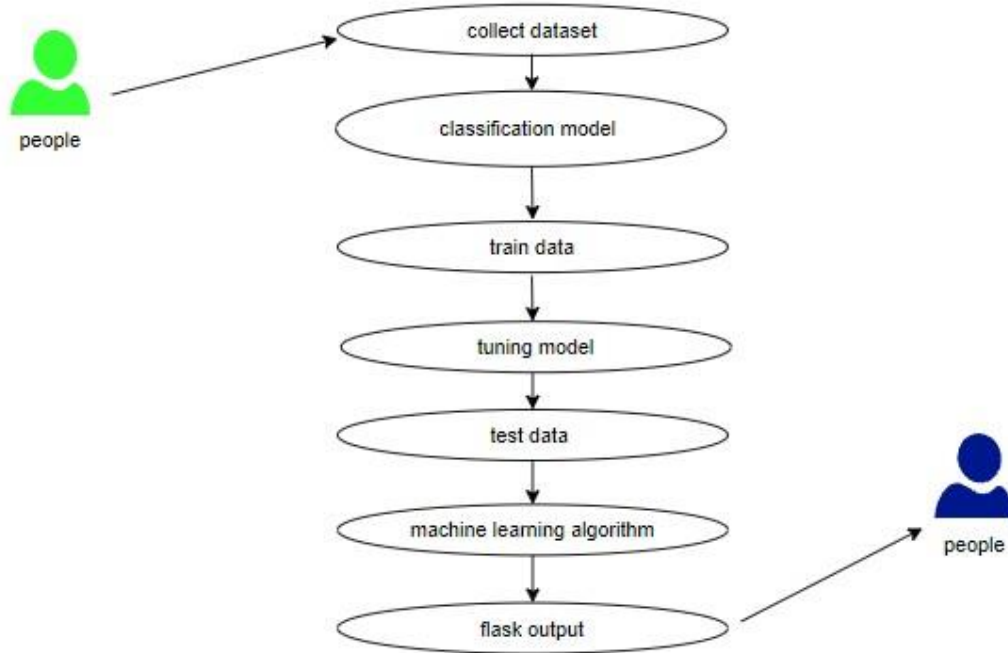


Fig 4.2 Use Case Diagram

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

4.3 Class Diagram:

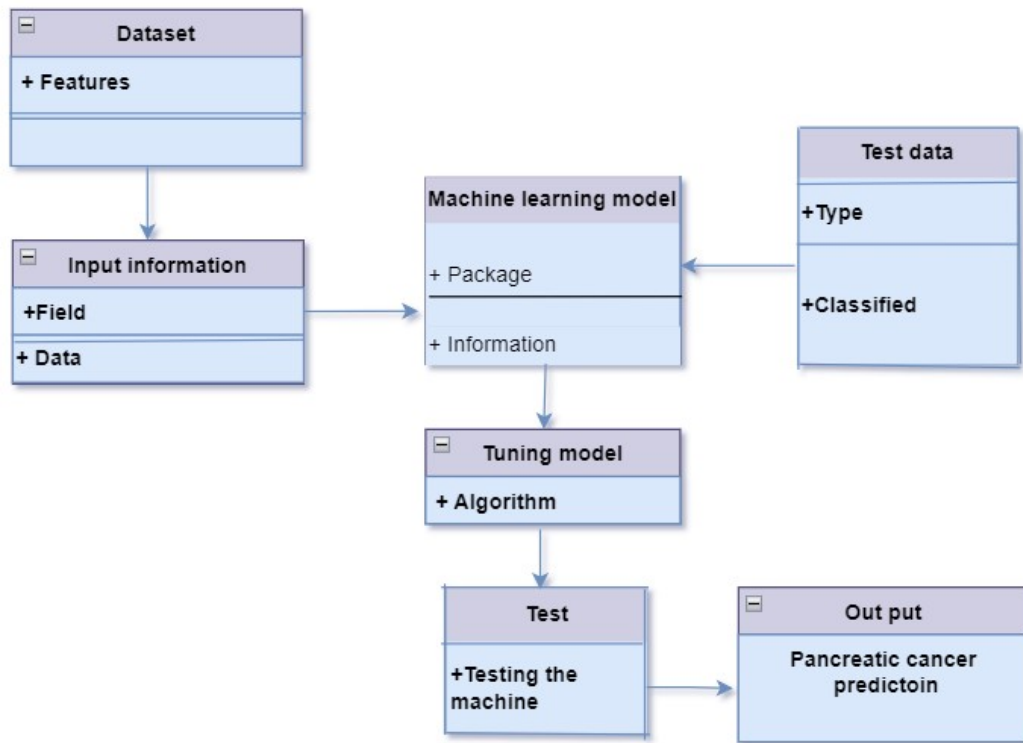


Figure 4.3 Class Diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

4.4 Activity Diagram:

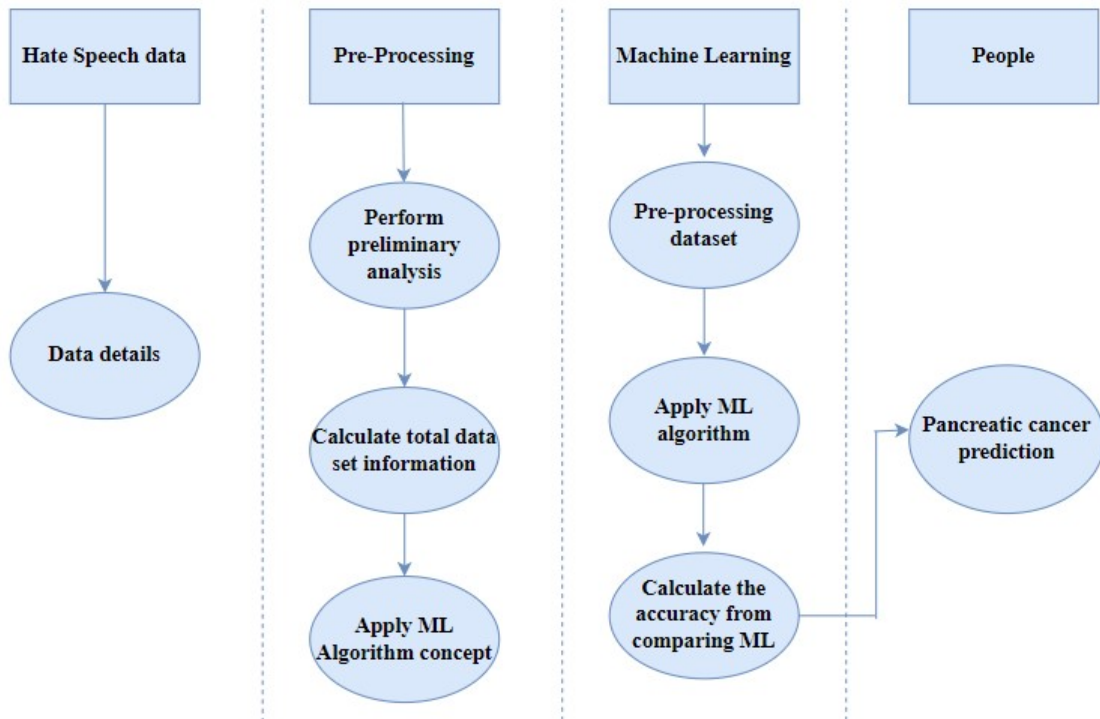


Fig 4.4 Activity Diagram

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

4.5 Sequence Diagram:

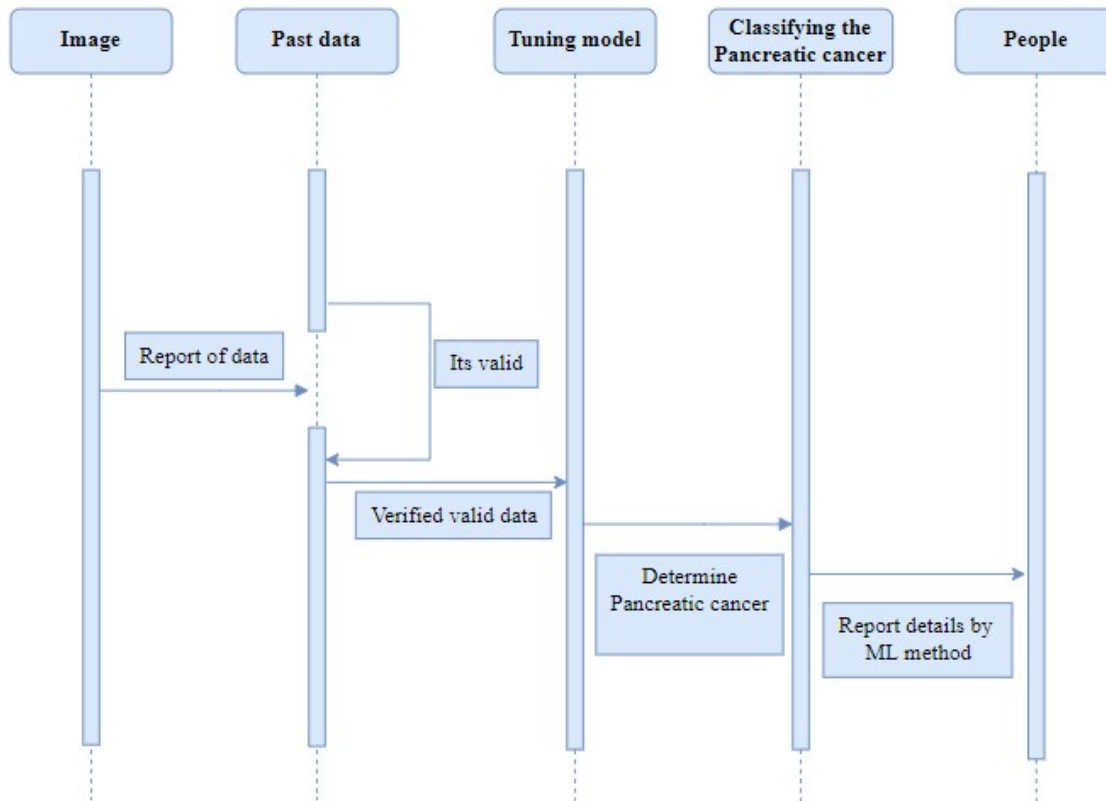


Figure 4.5 Sequence Diagram

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML 33rtefact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

4.6 Entity Relationship Diagram (ERD)

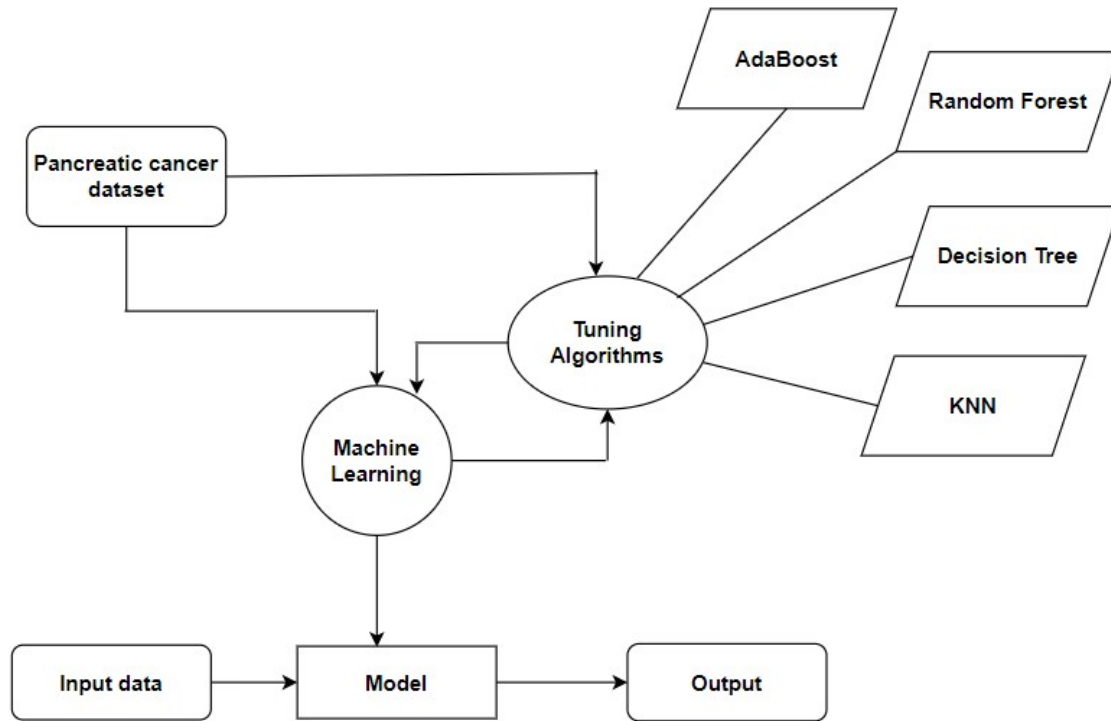


Fig 4.6 Entity Relationship Diagram (ERD)

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

CHAPTER 5

SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

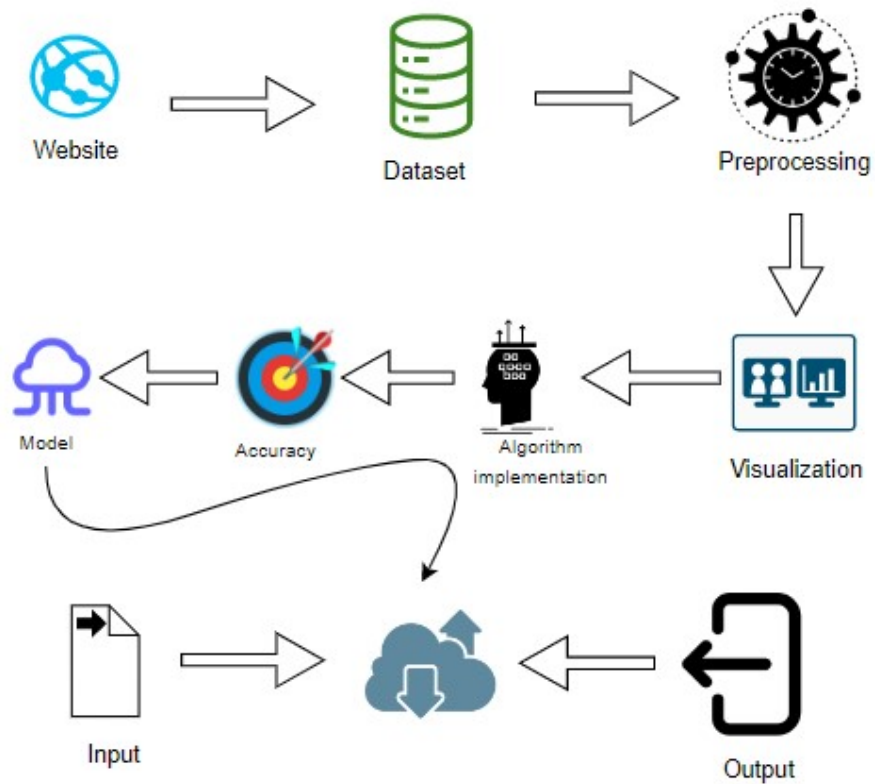


Fig 5.1 System Architecture

5.1 MODULES

- Data Pre-processing
- Data Analysis of Visualization
- Implementing Algorithm 1
- Implementing Algorithm 2
- Implementing Algorithm 3
- Implementing Algorithm 4

5.1.1 Module description

Data Pre-processing

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

Data Validation/ Cleaning/Preparing Process

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning models and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

MODULE DIAGRAM



Fig 5.2 Data Pre-processing Module Diagram

GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

Exploration data analysis of visualization

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

MODULE DIAGRAM



Fig 5.3 Data Visualization Module Diagram

GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple

different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

- KNN
- Random Forest
- Decision Tree
- AdaBoost

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy

then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labelled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$\text{F-Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

F1-Score Formula:

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

5.2 ALGORITHM AND TECHNIQUES

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labelled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabelled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

K-Nearest Neighbour

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

MODULE DIAGRAM

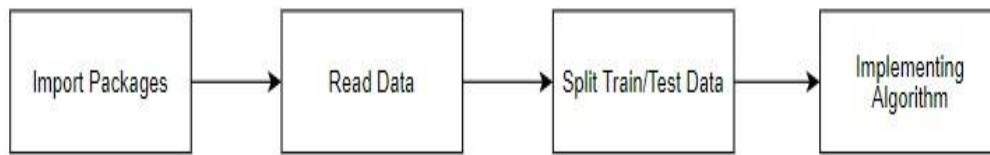


Fig 5.4 K-Nearest Neighbour Module Diagram

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

MODULE DIAGRAM

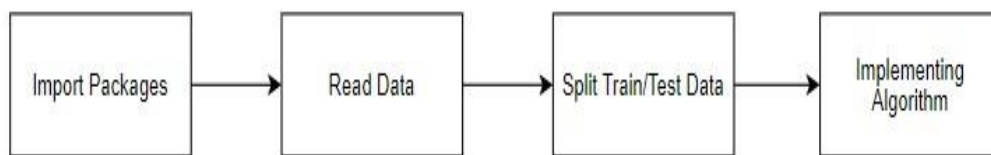


Fig 5.5 Random Forest Classifier Module Diagram

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

AdaBoost

This article was published as a part of the Data Science Blogathon

AdaBoost Algorithm

Introduction

Boosting is an ensemble modelling technique that was first presented by Freund and Schapire in the year 1997, since then, Boosting has been a prevalent technique for tackling binary classification problems. These algorithms improve the prediction power by converting a number of weak learners to strong learners.

The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly.

Let's take an example to understand this, suppose you built a decision tree algorithm on the Titanic dataset and from there you get an accuracy of 80%. After this, you apply a different algorithm and check the accuracy and it comes out to be 75% for KNN and 70% for Linear Regression.

We see the accuracy differs when we built a different model on the same dataset. But what if we use combinations of all these algorithms for making the final prediction? We'll get more accurate results by taking the average of results from these models. We can increase the prediction power in this way.

Boosting algorithms works in a similar way, it combines multiple models (weak learners) to reach the final output (strong learners).

MODULE DIAGRAM

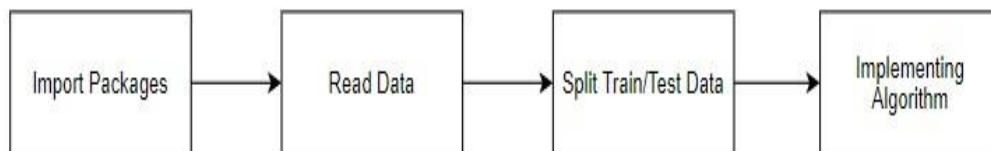


Fig 5.6 AdaBoost Module Diagram

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Decision Tree

Decision Tree is a Supervised Machine Learning Algorithm that uses a set of rules to make decisions, similarly to how humans make decisions.

One way to think of a Machine Learning classification algorithm is that it is built to make decisions.

You usually say the model predicts the class of the new, never-seen-before input but, behind the scenes, the algorithm has to decide which class to assign.

Some classification algorithms are probabilistic, like Naive Bayes, but there's also a rule-based approach.

We humans, also make rule-based decisions all the time.

When you're planning your next vacation, you use a rule-based approach. You might pick a different destination based on how long you're going to be on vacation, the budget available or if your extended family is coming along.

The answer to these questions informs the final decision. And if you continually narrow down the available vacation destinations based on how you answer each question, you can visualize this decision process as a (decision) tree.

MODULE DIAGRAM

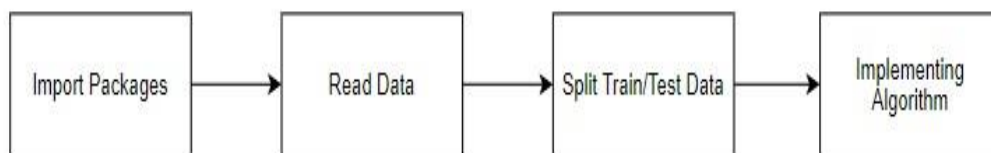


Fig 5.7 Decision Tree Module Diagram

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

5.3 Deployment

Flask (Web Frame Work)

Flask is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Flask is part of the Pallets Projects, and based on several others of them.

Flask is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained

popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python.

FEATURES

Flask was designed to be **easy to use and extend**. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to **plug in any extensions** you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Flask as your web application framework if we have immensely powerful Django, Pyramid, and don't forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Flask is pretty impressive too with it's:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus Flask gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down

- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.
- It is super easy to deploy Flask in production (Flask is 100%_WSGI 1.0 compliant")
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Flask is one of the most polished and feature-rich micro frameworks, available. Still young, Flask has a thriving community, first-class extensions, and an **elegant API**. Flask comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Flask.

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Flask is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Flask is a good start if you are getting into web development. Flask is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Flask

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from Python language. Flask provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework flask still doesn't use the Model View Controller (MVC) method.

Flask-RESTful is an extension for Flask that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Flask-Restful is a lightweight abstraction that works with the existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup.

Flask Restful is an extension for Flask that adds support for building REST APIs in Python using Flask as the back-end. It encourages best practices and is very easy to set up. Flask restful is very easy to pick up if you're already familiar with flask.

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

Start Using an API

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an `__init__.py` file inside) or a standard module (just a `.py` file).

For more information about resource loading, see `open_resource()`.

Usually you create a Flask instance in your main module or in the `__init__.py` file of your package.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to `view_func.__name__`.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
- **provide_automatic_options** (*Optional[bool]*) – Add the `OPTIONS` method and respond to `OPTIONS` requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.

Return type -- None

After_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use `teardown_request()` for that.

Parameters

f (*Callable[[Response], Response]*)

Return type

`Callable[[Response], Response]`

`after_request_funcs: t.Dict[AppOrBlueprintKey,`

`t.List[AfterRequestCallable]]`

A data structure of functions to call at the end of each request, in the format `{scope: [functions]}`. The `scope` key is the name of a blueprint the functions are active for, or `None` for all requests.

To register a function, use the `after_request()` decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

`app_context()`

Create an `AppContext`. Use as a `with` block to push the context, which will make `current_app` point at this application.

An application context is automatically pushed by `RequestContext.push()` when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With `app.app_context()`:

`init_db()`

CHAPTER 6

SYSTEM IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

6.1 Coding

Module – 1

Pre-Processing

```
# importing libraries
import pandas as pd
import numpy as np
# ignoring warnings

import warnings
warnings.filterwarnings('ignore')
# Reading the dataset

df=pd.read_csv('Debernardi et al 2020 data.csv')
df.head(20)
# Total rows and columns

print('Total rows: {}\nTotal columns: {}'.format(df.shape[0],df.shape[1]))
# data types of each feature

df.info()
# Checking null values

df.isnull().sum()
# Checking duplicated records
```

```

print('duplicated records: ',df.duplicated().sum())
# categorical column vs numerical column

cat_columns=[feature for feature in df.columns if df[feature].dtypes=='O']
num_columns=[feature for feature in df.columns if df[feature].dtypes!='O']

print('Total categorical columns: ',len(cat_columns))
print('Total numerical columns: ',len(num_columns))
# Unique features in numerical columns

for feature in cat_columns:
    print(feature+' :',len(df[feature].unique()))
# Numerical columns

print(num_columns)
# Describe function for numerical columns

df.describe()
# Correlation between the numerical columns
df.corr()

```

DATA PREPROCESSING

```

# Removing null values

df.isnull().sum()
# Since the null values are more
# Removing those columns

```

```
df.drop(columns=['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A'],axis=1,inplace=True)
```

```
df.head()
```

```
# Shape after removing columns with null values
```

```
df.shape
```

```
# Using label encoding
```

```
# it transforms the values into 0 and 1
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
for feature in cat_columns:
```

```
    if feature not in
```

```
['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A']:
```

```
    df[feature]=le.fit_transform(df[feature])
```

```
df.head()
```

Module – 2

Visualization

```
# importing libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```



```
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

```
# ignoring warnings
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
# Reading the dataset
```

```
df=pd.read_csv('Debernardi et al 2020 data.csv')
```

```
df.head()
```

```
# Dropping null values before doing visualization
```

```
# Removing null values columns
```

```
df.drop(columns=['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A'],axis=1,inplace=True)
```

```
df.head()
```

Data visualization

1. Checking dataset is balanced or not using pie chart

```
pieplot=df['diagnosis'].value_counts()
```

```
ax=pieplot.plot.pie(figsize=(12,6), autopct='%1.2f%%',  
explode=[0.01,0.05,0.1], shadow=True)
```

2. Comparing diagnosis with sex

```
sns.countplot(x=df['diagnosis'],hue=df['sex'])
```

3. Age with diagnosis

```
sns.violinplot(x=df['diagnosis'],y=df['age'])
```

4. Sample origin vs diagnosis

```
sns.countplot(x=df['sample_origin'],hue=df['diagnosis'])
```

5. creatinine vs diagnosis

```
sns.histplot(x=df['creatinine'],hue=df['diagnosis'])
```

6. Pair plot

```
sns.pairplot(df)
```

Module – 3

K nearest neighbour

In []:

```
# Importing the libraries  
  
import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Ignoring the values  
  
import warnings  
warnings.filterwarnings('ignore')
```

In []:

In []:

```
# Reading the dataset

df=pd.read_csv('Debernardi et al 2020 data.csv')
df.head()
```

In []:

```
# Shape

print('Shape: ',df.shape)
```

In []:

```
# Checking null values

df.isnull().sum()
```

In []:

```
# removing null values columns
# and also dropping sample_id column since id is not important

df.drop(columns=['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A',
'sample_id'],inplace=True)
df.head()
```

In []:

```
# Checking duplicated values

print('duplicated: ',df.duplicated().sum())
```

In []:

```
# Using label encoder on target feature

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
df['patient_cohort']=le.fit_transform(df['patient_cohort']).astype(int)
df['sample_origin']=le.fit_transform(df['sample_origin']).astype(int)
df['sex']=le.fit_transform(df['sex']).astype(int)
df.head()
```

In []:

```
# shape of dataset after ohe

df.shape
```

In []:

```
# Splitting the data into x and y

X=df.drop(columns='diagnosis', axis=1)
y=df.loc[:, 'diagnosis']

X.shape, y.shape
```

MODEL DEVELOPMENT

In []:

```
### Train test split
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,  
random_state=0)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

In []:

```
# k nearest neighbour
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier()
```

```
knn.fit(X,y)
```

In []:

```
# Y predict
```

```
y_predict=knn.predict(X_test)
```

In []:

```
# metrics
```

```
from sklearn.metrics import accuracy_score, classification_report,  
confusion_matrix
```

```
accuracy=accuracy_score(y_test,y_predict)
```

```
cr=classification_report(y_test,y_predict)
```

```
cm=confusion_matrix(y_test,y_predict)
```

```
print('accuracy: {} \n \n \n classification report: \n {} \n \n \n confusion  
matrix: \n {}'.format(accuracy*100,cr,cm))
```

In []:

```
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.cool):
```

```
    plt.imshow(cm, interpolation='nearest')
```

```
    plt.title(title)
```

```
    plt.colorbar()
```

```
cm1=confusion_matrix(y_test, y_predict)
```

```
print('Confusion matrix:')
```

```
print(cm)
```

```
plot_confusion_matrix(cm)
```

In []:

In []:

In []:

Module – 4

DECISION TREE

In []:

```
# Importing the libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In []:

```
# Ignoring the values

import warnings
warnings.filterwarnings('ignore')
```

In []:

```
# Reading the dataset

df=pd.read_csv('Debernardi et al 2020 data.csv')
df.head()
```

In []:

```
# Shape

print('shape: ',df.shape)
```

In []:

```
# Checking null values

df.isnull().sum()
```

In []:

```
# removing null values columns
# and also dropping sample_id column since id is not important

df.drop(columns=['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A',
'sample_id'],inplace=True)
df.head()
```

In []:

```
df['diagnosis'].value_counts()
```

In []:

```
# Checking duplicated values

print('duplicated: ',df.duplicated().sum())
```

In []:

```
# Using label encoder on target feature
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()  
df['patient_cohort']=le.fit_transform(df['patient_cohort']).astype(int)  
df['sample_origin']=le.fit_transform(df['sample_origin']).astype(int)  
df['sex']=le.fit_transform(df['sex']).astype(int)  
df.head()
```

In []:

```
df.shape
```

In []:

```
# Splitting the data into x and y
```

```
X=df.drop(columns='diagnosis', axis=1)  
y=df.loc[:, 'diagnosis']
```

```
X.shape, y.shape
```

MODEL DEVELOPMENT

In []:

```
### Train test split
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,  
random_state=42)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

In []:

```
## Decision Tree
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()  
dt.fit(X,y)
```

In []:

```
# y_predict
```

```
y_predict=dt.predict(X_test)
```

In []:

```
# metrics
```

```
from sklearn.metrics import accuracy_score, classification_report,  
confusion_matrix
```

```
accuracy=accuracy_score(y_test,y_predict)  
cr=classification_report(y_test,y_predict)  
cm=confusion_matrix(y_test,y_predict)
```

```
print('accuracy: {}'.format(accuracy*100,cr,cm))
matrix:\n {}'.format(accuracy*100,cr,cm))
```

In []:

```
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()
```

```
cm1=confusion_matrix(y_test, y_predict)
print('Confusion matrix:')
print(cm)
plot_confusion_matrix(cm)
```

In []:

```
import joblib
```

```
joblib.dump(dt, 'DT.pkl')
```

In []:

Module – 5

Random Forest

In []:

```
# Importing the libraries
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In []:

```
# Ignoring the values
```

```
import warnings
warnings.filterwarnings('ignore')
```

In []:

```
# Reading the dataset
```

```
df=pd.read_csv('Debernardi et al 2020 data.csv')
df.head()
```

In []:

```
# Shape
```

```
print('shape: ',df.shape)
```

In []:

```
# Checking null values
```

```

df.isnull().sum()
In [ ]:

# removing null values columns
# and also dropping sample_id column since id is not important

df.drop(columns=['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A',
'sample_id'],inplace=True)
df.head()
In [ ]:

df['diagnosis'].value_counts()
In [ ]:

# Checking duplicated values

print('duplicated: ',df.duplicated().sum())
In [ ]:

# Using label encoder on target feature

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
df['patient_cohort']=le.fit_transform(df['patient_cohort']).astype(int)
df['sample_origin']=le.fit_transform(df['sample_origin']).astype(int)
df['sex']=le.fit_transform(df['sex']).astype(int)
df.head()
In [ ]:

df.shape
In [ ]:

# Splitting the data into x and y

X=df.drop(columns='diagnosis', axis=1)
y=df.loc[:,'diagnosis']

X.shape, y.shape
In [ ]:

X.head()
In [ ]:

y.head()

```

MODEL DEVELOPMENT

```

In [ ]:

### Train test split

from sklearn.model_selection import train_test_split

```



```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,
random_state=42)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

In []:

```
## Decision Tree
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
dt=RandomForestClassifier()
dt.fit(X,y)
```

In []:

```
# y_predict
```

```
y_predict=dt.predict(X_test)
```

In []:

```
# metrics
```

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

```
accuracy=accuracy_score(y_test,y_predict)
cr=classification_report(y_test,y_predict)
cm=confusion_matrix(y_test,y_predict)
```

```
print('accuracy: {} \n \n classification report: \n {} \n \n confusion
matrix: \n {}'.format(accuracy*100,cr,cm))
```

In []:

```
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()
```

```
cm1=confusion_matrix(y_test, y_predict)
print('Confusion matrix:')
print(cm)
plot_confusion_matrix(cm)
```

Module – 6

ADABOOST CLASSIFIER

In []:

```
# Importing the libraries
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In []:

```
# Ignoring the values
```

```
import warnings
warnings.filterwarnings('ignore')
```

In []:

```
# Reading the dataset
```

```
df=pd.read_csv('Debernardi et al 2020 data.csv')
df.head()
```

In []:

```
# Shape
```

```
print('Shape: ',df.shape)
```

FEATURE ENGINEERING

In []:

```
# Checking null values
```

```
df.isnull().sum()
```

In []:

```
# removing null values columns
# and also dropping sample_id column since id is not important
```

```
df.drop(columns=['stage','benign_sample_diagnosis','plasma_CA19_9','REG1A',
'sample_id'],inplace=True)
df.head()
```

In []:

```
# Checking duplicated values
```

```
print('duplicated: ',df.duplicated().sum())
```

In []:

```
# Using label encoder on target feature
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
df['patient_cohort']=le.fit_transform(df['patient_cohort']).astype(int)
df['sample_origin']=le.fit_transform(df['sample_origin']).astype(int)
df['sex']=le.fit_transform(df['sex']).astype(int)
df.head()
```

In []:

```
# shape of dataset after ohe
```

```
df.shape
```

In []:

```
# Splitting the data into x and y

X=df.drop(columns='diagnosis', axis=1)
y=df.loc[:, 'diagnosis']

X.shape, y.shape
```

MODEL DEVELOPMENT

In []:

```
### Train test split

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,
random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

In []:

```
# Adaboost classifier

from sklearn.ensemble import AdaBoostClassifier

abc=AdaBoostClassifier()
abc.fit(X,y)
```

In []:

```
# y_predict

y_predict=abc.predict(X_test)
```

In []:

```
# metrics

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

accuracy=accuracy_score(y_test,y_predict)
cr=classification_report(y_test,y_predict)
cm=confusion_matrix(y_test,y_predict)

print('accuracy: {} \n \n classification report: \n {} \n \n confusion
matrix: \n {}'.format(accuracy*100,cr,cm))
```

In []:

```
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, y_predict)
print('Confusion matrix:')
print(cm)
plot_confusion_matrix(cm)
```

CHAPTER 7

SYSTEM TESTING

7. SYSTEM TESTING

7.1 UNIT TESTING

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

7.2 INTEGRATION TESTING

It is defined to be set of interaction, all defined interaction among components need to be tested. The architecture and design can give the detail of interaction within the system and explain how they interact with each other. The types of integration testing are Top-Down testing, Bottom-Up testing, Bi- Directional testing and System integration.

TEST CASE ID	INPUT	OUTPUT	PASS/FAIL
Test case 1	Age: 33 Gender: Female Creatinine: 1.83222 LYVE1: 0.893219 REG1B: 52.94884 TFF1: 654.2822	Patient has no pancreatic disease	
	Age: 73 Gender: Female Creatinine: 0.40716 LYVE1: 5.794321 REG1B: 398.9187 TFF1: 398.9187	Patient has pancreatic cancer	
	Age: 33 Gender: Female Creatinine: 0.74646 LYVE1: 4.9695 REG1B: 59.362 TFF1: 489.465	Patient has benign hepatobiliary disease	

CHAPTER 8

CONCLUSION

8. CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the Pancreatic cancer of the patient.

8.1 Future Work

- Deploying the project in the cloud.
- To optimize the work to implement in the IOT system.

APPENDIX

APPENDIX

9. Output Sample Screenshot

URINARY BIOMARKER FOR PANCREATIC CANCER PREDICTION

PATIENT_COHORT: Previously Used Sample (Cohort_1)

CREATININE: CREATININE

SAMPLE_ORGIN: Barts Pancreas Tissue Bank, London, UK (BPTB)

YVLE1: YVLE1

AGE: Enter your age

GENDER: MALE

REG1B: REG1B

TFF1: TFF1

SYMP TOMS

Figure 9.1 Sample Output Screen

PATIENT_COHORT: Previously Used Sample (Cohort_1)

CREATININE: 0.40716

SAMPLE_ORGIN: Barts Pancreas Tissue Bank, London, UK (BPTB)

YVLE1: 5.794321

AGE: 73

REG1B: 398.9187

GENDER: FEMALE

TFF1: 398.9187

SYMP TOMS

Figure 9.2 Sample Output Screen

Predict

Patient has
Pancreatic cancer

Figure 9.3 Sample Output Screen

REFERENCES

REFERENCES

1. Al Bakir, I., Curtius, K., Graham, T. A., & McDonald, S. A. C. (2019). Predicting pancreatic cancer risk using machine learning techniques. *Gut*, 68(2), 440-441.
2. Bustin, S. A. (2019). A guide to the technology and its applications for quantifying gene expression by real-time quantitative PCR. In *A-Z of quantitative PCR* (pp. 1-9). Springer, Cham.
3. Chen, Y., Shen, H., & Zhu, X. (2020). Machine learning in cancer biomarker discovery. *Epigenomics*, 12(10), 881-895.
4. Cho, Y., Kim, J. H., Kim, Y. H., Song, M., Lee, H. W., Kim, K. P., ... & Hwang, D. W. (2020). Urinary biomarkers for pancreatic ductal adenocarcinoma identified via proteomic analysis. *Scientific Reports*, 10(1), 1-9.
5. De Oliveira, C. P., Stefani, J. P., & Gimenez, I. F. (2020). The role of machine learning techniques in pancreatic cancer prediction. *Journal of Personalized Medicine*, 10(2), 56.
6. Dong, Y., Skelton, R., Ventura, I. A., Segal, M., & Gilbertson, J. R. (2021). A machine learning approach for predicting pancreatic cancer using urinary biomarkers. *Journal of Medical Systems*, 45(2), 1-7.
7. Ferreira, M. A., Souto, E. B., & Teixeira, J. A. (2019). Bioinformatics and machine learning in the development of urinary biomarkers of prostate cancer. *Trends in Biotechnology*, 37(3), 267-279.
8. He, Y., Deng, S., Wang, S., & Shi, Y. (2020). Early diagnosis of pancreatic cancer using machine learning and serum biomarkers. *Journal of Healthcare Engineering*, 2020, 1-11.
9. Isayeva, T., Imangaliyev, S., Kim, J. H., Cho, Y., Kim, Y. H., & Hwang, D. W. (2020). Urinary biomarkers for pancreatic cancer detection: current status and future perspectives. *Cancers*, 12(6), 1526.
10. Kalimuthu, S. N., Prasad, R., & Ramanathan, A. (2021). Machine learning and urinary biomarkers for pancreatic cancer detection. *Current Pharmaceutical Design*, 27(14), 1644-1652.