CANCER DETECTION USING MACHINE LEARNING

A Minor Project Report

Submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

By

YELURI BHARATH RAJ (20U01031)

PULI PRADEEP (20U01053)

AMGOTHU JAYARAM NAIK (20U01033)

BANALA ABHISHEK SATWIK (20U01027)

Supervisor:

Dr. Akhilesh Panchal

Department of Electronics and Communication Engineering

Indian Institute of Information Technology

Bhopal-462003 (India)

Certificate

Department of Electronics and Communication Engineering
Indian Institute of Information Technology Bhopal

It is certified that the work contained in the project report entitled "**CANCER DETECTION USING MACHINE LEARNING**" by the following students has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

YELURI BHARATH RAJ (20U01031)

PULI PRADEEP (20U01053)

AMGOTHU JAYARAM NAIK (20U01033)

BANALA ABHISHEK SATWIK (20U01027)

Date: Supervisor (Name & Signature)

This project report entitled "**CANCER DETECTION USING MACHINE LEARNING**" submitted by the group is approved for the degree of Bachelor of Technology.

The viva-voce examination has been held on _____

Supervisor(s) Examiner(s)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We declare that we have properly and accurately acknowledged all sources used in the production of this report. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and can also evoke penalty action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

YELURI BHARATH RAJ (20U01031) Signature:

PULI PRADEEP (20U01053) Signature:

AMGOTHU JAYARAM NAIK (20U01033) Signature:

BANALA ABHISHEK SATWIK (20U01027) Signature:

Acknowledgments

With due respect, we express our deep sense of gratitude to our respected supervisors

Dr. Akhilesh Panchal, for his invaluable support and guidance. We are thankful for the encouragement that she/he has given us in completing this project successfully. Her/his rigorous evaluation and constructive criticism were of great assistance. It is imperative for us to mention the fact that this minor project could not have been without the periodic suggestion and advice of our project.

We are also grateful to our respected Director ***Dr. Rama. S. Verma*** for permitting us to utilize all the necessary facilities of the college. Needless to mention is the additional help and support extended by our respected Nodal Officer***, Dr. Meenu Chawla***, in allowing us to use the department laboratories and other services.

We are also thankful to professor in charge examination ***Dr. Dheeraj K. Agrawal*** who continuously supported and encouraged us by his advices and also helped us in our project presentation that has improved our presentation skills. We are also thankful to professor in charge academics ***Dr. Amit Ojha*** who continuously supported and encouraged us by his advices and also helped us in our project presentation that has improved our presentation skills.

We are also thankful to Student Welfare in charge ***Dr. Jaytrilok Choudhary*** for constantly motivating us to work harder for the completion of the project. We extend our sincere thanks to ***Dr. Amit Bhagat*** who co-operated with us nicely for smooth development of this project. We would also like to thanks all the other faculty, staff members and laboratory attendants of our department for their kind co-operation and help. Last but certainly not the least, we would like to express our deep appreciation towards our family members and batch mates for providing the much-needed support and encouragement.

Personally, Thanks to All!

YELURI BHARATH RAJ (20U01031)

PULI PRADEEP (20U01053)

AMGOTHU JAYARAM NAIK (20U01033)

BANALA ABHISHEK SATWIK (20U01027)

ABSTRACT

Cancer has been characterized as a heterogeneous disease consisting of many different subtypes. The early diagnosis and prognosis of a cancer type have become a necessity in cancer research, as it can facilitate the subsequent clinical management of patients. The importance of classifying cancer patients into high or low risk groups has led many research teams, from the biomedical and the bioinformatics field, to study the application of machine learning (ML) methods. Therefore, these techniques have been utilized as an aim to model the progression and treatment of cancerous conditions. In addition, the ability of ML tools to detect key features from complex datasets reveals their importance.

A variety of these techniques, including Artificial Neural Networks (ANNs), Bayesian Networks (BNs), Support Vector Machines (SVMs) and Decision Trees (DTs) have been widely applied in cancer research for the development of predictive models, resulting in effective and accurate decision making. Even though it is evident that the use of ML methods can improve our understanding of cancer progression, an appropriate level of validation is needed in order for these methods to be considered in the everyday clinical practice. In this work, we present a review of recent ML approaches employed in the modeling of cancer progression. The predictive models discussed here are based on various supervised ML techniques as well as on different input features and data samples. Given the growing trend on the application of ML methods in cancer research, we present here the most recent publications that employ these techniques as an aim to model cancer risk or patient outcomes.

TABLE OF CONTENTS

INTRODUCTION

Over the past decades, a continuous evolution related to cancer research has been performed. Scientists applied different methods, such as screening in early stage, in order to find types of cancer before they cause symptoms. Moreover, they have developed new strategies for the early prediction of cancer treatment outcome. With the advent of new technologies in the field of medicine, large amounts of cancer data have been collected and are available to the medical research community. However, the accurate prediction of a disease outcome is one of the most interesting and challenging tasks for physicians. As a result, ML methods have become a popular tool for medical researchers. These techniques can discover and identify patterns and relationships between them, from complex datasets, while they are able to effectively predict future outcomes of a cancer type. Given the significance of personalized medicine and the growing trend on the application of ML techniques, we here present a review of studies that make use of these methods regarding the cancer prediction and prognosis. In these studies prognostic and predictive features are considered which may be independent of a certain treatment or are integrated in order to guide therapy for cancer patients, respectively. In addition, we discuss the types of ML methods being used, the types of data they integrate, the overall performance of each proposed scheme while we also discuss their pros and cons. An obvious trend in the proposed works includes the integration of mixed data, such as clinical and genomic. However, a common problem that we noticed in several works is the lack of external validation or testing regarding the predictive performance of their models. It is clear that the application of ML methods could improve the accuracy of cancer susceptibility, recurrence and survival prediction. The accuracy of cancer prediction outcome has significantly improved by 15%–20% the last years, with the application of ML techniques. Several studies have been reported in the

literature and are based on different strategies that could enable the early cancer diagnosis and prognosis. Specifically, these studies describe approaches related to the profiling of circulating miRNAs that have been proven a promising class for cancer detection and identification. However, these methods suffer from low sensitivity regarding their use in screening at early stages and their difficulty to discriminate benign from malignant tumours. Various aspects regarding the prediction of cancer outcome based on gene expression signatures are discussed. These studies list the potential as well as the limitations of microarrays for the prediction of cancer outcome. Even though gene signatures could significantly improve our ability for prognosis in cancer patients, poor progress has been made for their application in the clinics. However, before gene expression profiling can be used in clinical practice, studies with larger data samples and more adequate validation are needed. In the present work only studies that employed ML techniques for modelling cancer diagnosis and prognosis are presented.

METHODOLOGY & WORK DESCRIPTION

DATA SET

The data used for the experiments was acquired from Kaggle. This dataset is BreakHist Dataset consisting of four directories representing the magnification of the images respectively i.e. 100X, 200X, 400X and 40X. The dataset consists of 7,858 instances in total which are divided into the four magnification directories. Each magnification directory consists of two directories representing the tumours i.e. Benign and Malignant.

PREPROCESSING

Feature Selection

The importance of feature selection in a machine learning model is inevitable. It turns the data to be free from ambiguity and reduces the complexity of the data. Also, it reduces the size of the data, so it is easy to train the model and reduces the training time. It avoids over fitting of data. Selecting the best feature subset from all the features increases the accuracy. Some feature selection methods are wrapper methods, filter methods, and embedded methods.

Recursive Feature Elimination

RFE is a wrapper-type feature selection algorithm. This means that a different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. This is in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score. Technically, RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally. RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains. This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model.

PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modelling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation system, optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable

CROSS-VALIDATION

In machine learning, we couldn't fit the model on the training data and can't say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the cross-validation technique.

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance.

The main purpose of cross validation is to prevent over fitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model's generalization performance, i.e., its ability to perform well on new, unseen data.

There are several types of cross validation techniques, including k-fold cross validation, leave-one-out cross validation, and stratified cross validation. The choice of technique depends on the size and nature of the data, as well as the specific requirements of the modelling problem.

Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set. The three steps involved in cross-validation are as follows:

1. **Reserve some portion of sample data-set.**
2. **Using the rest data-set train the model.**
3. **Test the model using the reserve portion of the data-set.**

PROPOSED ALGORITHM

NAIVE BAYES ALGORITHM

```
def mock_test(data):
inputs=np.array(data)
malignant_probability=s.multivariate_normal.pdf(inputs,mu_hat,sigma_hat)*malignant_prior
benign_probability=s.multivariate_normal.pdf(inputs,mu_hat_b,sigma_hat_b)*benign_prior
boolean_mask= malignant_probability>benign_probability
predict_category= pd.Series(boolean_mask)
predict_category.replace(to_replace=[True,False],value=['M','B'],inplace=True)
return np.array(predict_category)
```

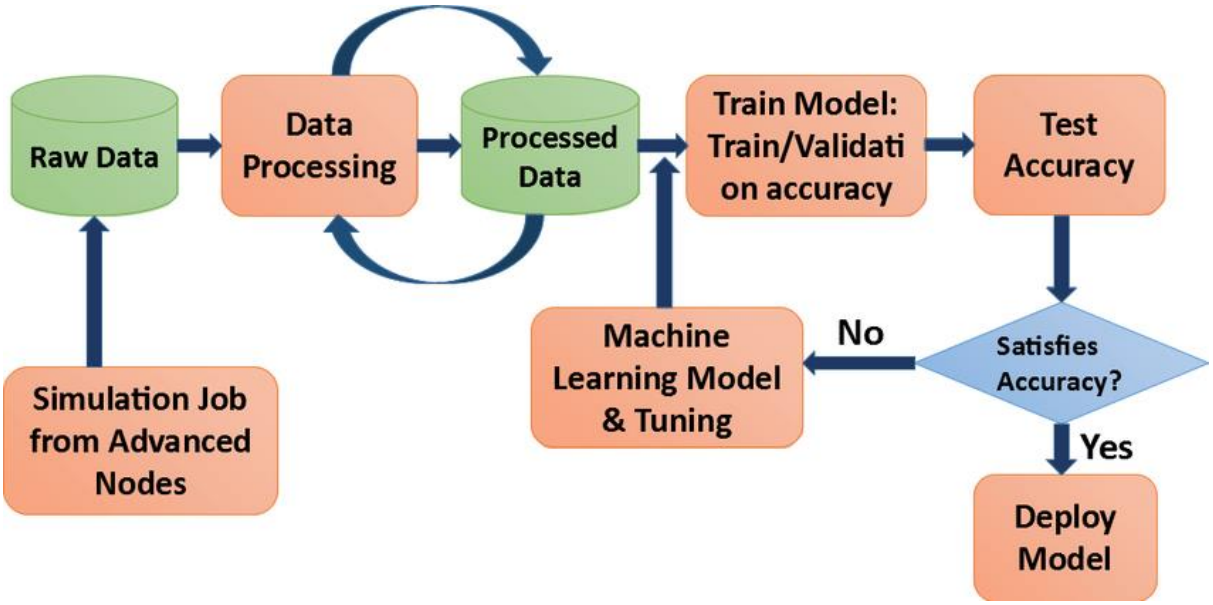
NAIVE BAYES

Naïve Bayes classifier is a supervised learning algorithm that is used for classification. It is based on the Bayes theorem that is finding the probability of an event after an event has already occurred. It is one of the simplest yet powerful ML algorithms in use and finds applications in many industries. Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases. This algorithm faces the ‘zero frequency problem’ where it assigns zero probability to a categorical variable whose category in the test data set wasn’t available in the training dataset. We can overcome this issue by using a smoothing technique. Also, another drawback of Naïve Bayes is that it requires large data sets to attain its best accuracy**.**

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
It is mainly used in text classification that includes a high-dimensional training dataset.

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

PROPOSED FLOWCHART/BLOCK DIAGRAM



TECHNOLOGY & PLATFORM USED

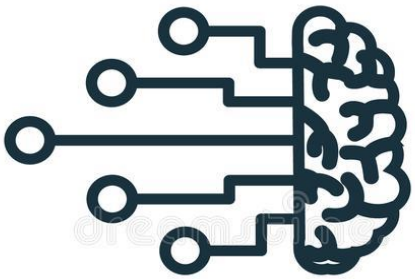
Technology used: Python**,** Machine Learning

Platform: Google colab

TECHNOLOGY USED

Python is a dynamically-typed, high-level programming language designed to be easy to read and write. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Some of its key features include: Readability: Python's syntax is designed to be highly readable and can be written with fewer lines of code than many other programming languages. Dynamic Typing: Python uses dynamic typing, meaning that you do not have to specify the type of a variable before using it.

This makes the code more flexible but can also result in runtime errors if not used carefully. Large Standard Library: Python has a large standard library that includes modules for file I/O, string manipulation, data compression, and many other common tasks. Cross-Platform Compatibility: Python can run on many different operating systems, including Windows, Mac, and Linux. Large Community: Python has a large and active community of developers, who have created many libraries, tools, and frameworks that make development with Python easier and more efficient. These features, among others, make Python a popular choice for a wide range of applications, including web development, scientific computing, data analysis, machine learning, and more.



MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

IBM has a rich history with machine learning. One of its own, Arthur Samuel, is credited for coining the term, “machine learning” with his research (PDF, 481 KB) (link resides outside IBM) around the game of checkers. Robert Nealey, the self-proclaimed checkers master, played the game on an IBM 7094 computer in 1962, and he lost to the computer. Compared to what can be done today, this feat seems trivial, but it’s considered a major milestone in the field of artificial intelligence.

Over the last couple of decades, the technological advances in storage and processing power have enabled some innovative products based on machine learning, such as Netflix’s recommendation engine and self-driving cars.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications

and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

PLATFORM USED

Colaboratory ("Colab" for short) is a data analysis and machine learning tool that allows you to combine executable Python code and rich text along with charts, images, HTML, LaTeX and more into a single document stored in Google Drive. It connects to powerful Google Cloud Platform runtimes and enables you to easily share your work and collaborate with others.

IMPLEMENTATION & CODING

Implementation refers to the process of turning the project's design and plan into a working system or application. It involves writing code, integrating different components, testing the system, and deploying it.

The implementation phase is where the actual development work takes place. It is where the programming languages, libraries, frameworks, and tools are used to create a functional system that meets the requirements specified in the project plan. This phase requires a good understanding of the project's design and specifications and a strong command of programming languages and tools.

During implementation, we faced unexpected challenges, such as technical issues or changes in requirements. Therefore, it is crucial to have a well-planned and flexible development process that allows for adjustments and changes. It is good that we had a well-developed ecosystem for such challenges. As discussed earlier in methodology, we have organized acad-team, which prepares the problems and problems step-wise solution. After acad team, the work is transferred to the backend or Solver Team. The Backend Solver team implements or codes the category of the problems in solution template by the acad-team. The code created by the backend team is then check and report is generated by some condition failing inside the code

CODING:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as s
from sklearn.metrics import classification_report
```

```
data = pd.read_csv('/content/drive/MyDrive/data (1).csv')
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...

5 rows × 33 columns

```
[ ] data.columns
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

```
[ ] data.drop([data.columns[0],data.columns[32]],axis=1,inplace= True)
```

```
[ ] data.columns
```

```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

```
[ ] labels = np.array(data['diagnosis']).reshape(data['diagnosis'].shape[0],1)
    labels.shape
```

```
(569, 1)
```

```
[ ] X= np.array(data.iloc[:,1:])
    X.shape
```

```
(569, 30)
```

```
[ ] mu = np.mean(X,axis=0)
    mu.shape
```

```
(30,)
```

```
[ ] mu_hat=mu.reshape(1,mu.shape[0])
    mu_hat.shape
```

```
(1, 30)
```

```
[ ] X_dash=X-mu_hat
    X_dash.shape
```

```
(569, 30)
```

```
[ ] sigma_hat=1/(data.shape[0])*np.matmul(X_dash.T,X_dash)
    sigma_hat.shape
```

```
(30, 30)
```

```
[ ] #now do singular value decomposition o sigma hat
    sigma_hat_decompose = np.linalg.svd(sigma_hat)
    len(sigma_hat_decompose)
```

```
3
```

```
[ ] Q=sigma_hat_decompose[0]
    lamda=sigma_hat_decompose[1]
```

```
[ ] lamda
```

```
array([4.43002671e+05, 7.29725279e+03, 7.02596776e+02, 5.45526944e+01,
       3.98199123e+01, 2.99930721e+00, 1.81213991e+00, 3.70813899e-01,
       1.55240237e-01, 8.39134846e-02, 3.15534015e-02, 7.48418875e-03,
       3.15610001e-03, 2.15770517e-03, 1.32420744e-03, 6.39144051e-04,
       3.74224474e-04, 2.34756323e-04, 1.84259067e-04, 1.63891523e-04,
       7.79729248e-05, 5.75099161e-05, 3.48559114e-05, 2.83453651e-05,
       1.61179909e-05, 1.24682907e-05, 3.67401338e-06, 2.84289915e-06,
       2.00139207e-06, 7.00763523e-07])
```

```
[ ] Q_tilda =Q[:,0:15]
    Q_tilda.shape
```

```
(30, 15)
```

```
    X_new=np.matmul(X_dash,Q_tilda)
    X_new.shape
```

```
(569, 15)
```

```
[ ] new_data=pd.DataFrame(data=X_new)
```

```
[ ] new_data.head()
```

[ ]

	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	-1160.142574	-293.917544	48.578398	8.711975	32.000486	-1.265415	0.931337	-0.148167	-0.745463	0.589359	0.307804	-0.043452	0.034777	-0.0
1	-1269.122443	15.630182	-35.394534	-17.861283	-4.334874	0.225872	-0.046037	-0.200804	0.485828	-0.084035	-0.080642	-0.033042	0.045485	0.0
2	-995.793889	39.156743	-1.709753	-4.199340	-0.466529	2.652811	-0.779745	0.274026	0.173874	-0.186994	-0.279174	0.020464	0.083505	-0.0
3	407.180803	-67.380320	8.672848	11.759867	7.115461	-1.299436	-1.267304	0.060555	0.330639	-0.144155	-0.927471	0.174720	0.282556	-0.0
4	-930.341180	189.340742	1.374801	-8.499183	7.613289	-1.021160	-0.335522	-0.289109	-0.036087	-0.138502	-0.042228	0.062721	-0.114247	-0.0

```
[ ] new_data['diagnosis']=labels
    #this is our projectd data
    new_data.shape
```

```
(569, 16)
```

```
[ ] new_data
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	-1160.142574	-293.917544	48.578398	8.711975	32.000486	-1.265415	0.931337	-0.148167	-0.745463	0.589359	0.307804	-0.043452	0.034777	-
1	-1269.122443	15.630182	-35.394534	-17.861283	-4.334874	0.225872	-0.046037	-0.200804	0.485828	-0.084035	-0.080642	-0.033042	0.045485	
2	-995.793889	39.156743	-1.709753	-4.199340	-0.466529	2.652811	-0.779745	0.274026	0.173874	-0.186994	-0.279174	0.020464	0.083505	-
3	407.180803	-67.380320	8.672848	11.759867	7.115461	-1.299436	-1.267304	0.060555	0.330639	-0.144155	-0.927471	0.174720	0.282556	-
4	-930.341180	189.340742	1.374801	-8.499183	7.613289	-1.021160	-0.335522	-0.289109	-0.036087	-0.138502	-0.042228	0.062721	-0.114247	-

[ ]

564	-1414.126684	110.222492	40.065944	-6.562240	-5.102856	0.395424	-0.786751	-0.037082	0.452530	-0.235185	-0.163649	-0.052543	-0.075032	
565	-1045.018854	77.057589	0.036669	4.753245	-12.417863	0.059637	0.449831	-0.509154	0.449986	0.493247	-0.007625	-0.055832	-0.015163	-
566	-314.501756	47.553525	-10.442407	9.771881	-6.156213	0.870726	-2.166493	0.442279	0.097398	-0.144667	0.109147	-0.076263	-0.004448	
567	-1124.858115	34.129225	-19.742087	23.660881	3.565133	-4.086390	-1.705401	0.359964	-0.385030	0.615467	-0.307166	0.028224	0.060561	
568	771.527622	-88.643106	23.889032	-2.547249	-14.717566	-4.418123	-2.815752	-0.030039	0.423451	-0.301439	-0.133353	0.115105	-0.019667	-

569 rows × 16 columns

```
[ ] new_data[new_data['diagnosis']=='M'].shape
```

```
(212, 16)
```

```
new_data[new_data['diagnosis']=='B'].shape
```

```
(357, 16)
```

CROSS VALIDATION

```
[ ] training_data_len =int(0.7*new_data.shape[0])
    training_data_len
```

```
398
```

```
[ ] #we have to take training data lengths of  given classification are equally  otherwise it predicts wrongly
    benign_train_data=new_data[new_data['diagnosis']=='B'].iloc[0: int(training_data_len/2)]
    malignant_train_data=new_data[new_data['diagnosis']=='M'].iloc[0: int(training_data_len/2)]
```

```
[ ] benign_train_data.shape
```

```
(199, 16)
```

```
[ ] #now cancatenate both data
    train_data =pd.concat([benign_train_data,malignant_train_data],axis=0)
    train_data.shape
```

```
(398, 16)
```

```
[ ] remaining_data_of_bengign=new_data[new_data['diagnosis']=='B'].iloc[int(training_data_len/2):]
    remaining_data_of_bengign.shape
```

```
(158, 16)
```

```
    remaining_data_of_malignant=new_data[new_data['diagnosis']=='M'].iloc[int(training_data_len/2): ]
    remaining_data_of_malignant.shape
```

```
(13, 16)
```

```
[ ] #now we can take cross validation part of data
    cv_len =int(0.2*new_data.shape[0])
    cv_len
```

```
113
```

```
[ ] #now we can concatenate the remaining data
    remaining_data =pd.concat([remaining_data_of_bengign,remaining_data_of_malignant],axis=0)
    remaining_data.shape
```

```
(171, 16)
```

```
[ ] cross_validation_data= remaining_data.iloc[0:113, :]
    cross_validation_data.shape
```

```
(113, 16)
```

```
[ ] test_data =remaining_data.iloc[113: , :]
    test_data.shape
```

```
(58, 16)
```

```
[ ] test_data.tail()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	
563	-1167.136978	105.596679	11.102074	14.203503	7.357428	-2.824776	-3.130968	-0.961313	-0.789158	-0.675360	0.439485	-0.024826	-0.125908	-0
564	-1414.126684	110.222492	40.065944	-6.562240	-5.102856	0.395424	-0.786751	-0.037082	0.452530	-0.235185	-0.163649	-0.052543	-0.075032	0
565	-1045.018854	77.057589	0.036669	4.753245	-12.417863	0.059637	0.449831	-0.509154	0.449986	0.493247	-0.007625	-0.055832	-0.015163	-0
566	-314.501756	47.553525	-10.442407	9.771881	-6.156213	0.870726	-2.166493	0.442279	0.097398	-0.144667	0.109147	-0.076263	-0.004448	0
567	-1124.858115	34.129225	-19.742087	23.660881	3.565133	-4.086390	-1.705401	0.359964	-0.385030	0.615467	-0.307166	0.028224	0.060561	0

```
[ ] mu_hat =np.array(train_data[train_data['diagnosis'] =='M'].iloc[:,0:15].mean())
    sigma_hat=np.array(train_data[train_data['diagnosis']=='M'].iloc[:,0:15].cov())
```

```
[ ] mu_hat
```

```
[ ] array([-6.19149059e+02, -9.10604406e+00, -3.00251408e+00,  3.21867937e+00,
           6.62722209e-01, -1.49641755e-03,  1.33728677e-01, -6.56051800e-03,
           4.44266350e-02, -2.27845949e-02, -5.28254216e-02,  2.13821710e-03,
           2.13592658e-03, -4.87610367e-03,  1.12627757e-04])
```

```
[ ] np.linalg.det(sigma_hat)

    19723.27120644734
```

```
[ ] sigma_hat.shape

    (15, 15)
```

```
    #now same work for benign
    mu_hat_b=np.array(train_data[train_data['diagnosis']=='B'].iloc[:,0:15].mean())

    sigma_hat_b=np.array(train_data[train_data['diagnosis']=='B'].iloc[:,0:15].cov())
```

```
[ ] sigma_hat_b.shape

    (15, 15)
```

```
[ ] malignant_prior=0.5
    benign_prior=0.5
```

```
[ ] def mock_test(data):
        inputs=np.array(data)
        malignant_probability=s.multivariate_normal.pdf(inputs,mu_hat,sigma_hat)*malignant_prior
        benign_probability=s.multivariate_normal.pdf(inputs,mu_hat_b,sigma_hat_b)*benign_prior
        boolean_mask= malignant_probability>benign_probability
        predict_category= pd.Series(boolean_mask)
        predict_category.replace(to_replace=[True,False],value=['M','B'],inplace=True)
        return np.array(predict_category)
```

```
[ ] predicted_output=mock_test(cross_validation_data.iloc[:,0:15])
    actual_output=np.array(cross_validation_data['diagnosis'])
    boolean_mask=actual_output==predicted_output
```

```
[ ] accuracy =np.count_nonzero(boolean_mask)/boolean_mask.shape[0]
    accuracy

    0.9292035398230089
```

```
    #if we do on test_data
    test_pred =mock_test(test_data.iloc[:,0:15])
    test_pred
```

```
    array(['B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
           'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
           'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
           'B', 'B', 'B', 'B', 'B', 'B', 'M', 'M', 'M', 'M', 'M', 'M', 'B',
           'M', 'M', 'M', 'M', 'M', 'M'], dtype=object)
```

```
[ ] actual_test_result=np.array(test_data['diagnosis'])
    actual_test_result
```

```
[ ] array(['B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
           'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
           'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
           'B', 'B', 'B', 'B', 'B', 'B', 'M', 'M', 'M', 'M', 'M', 'M', 'M',
           'M', 'M', 'M', 'M', 'M', 'M'], dtype=object)
```

```
[ ] boolean_mask1= actual_test_result == test_pred
```

```
[ ] test_accuracy =np.count_nonzero(boolean_mask1)/boolean_mask1.shape[0]
    test_accuracy
```

```
0.9827586206896551
```

THE FINAL RESULT

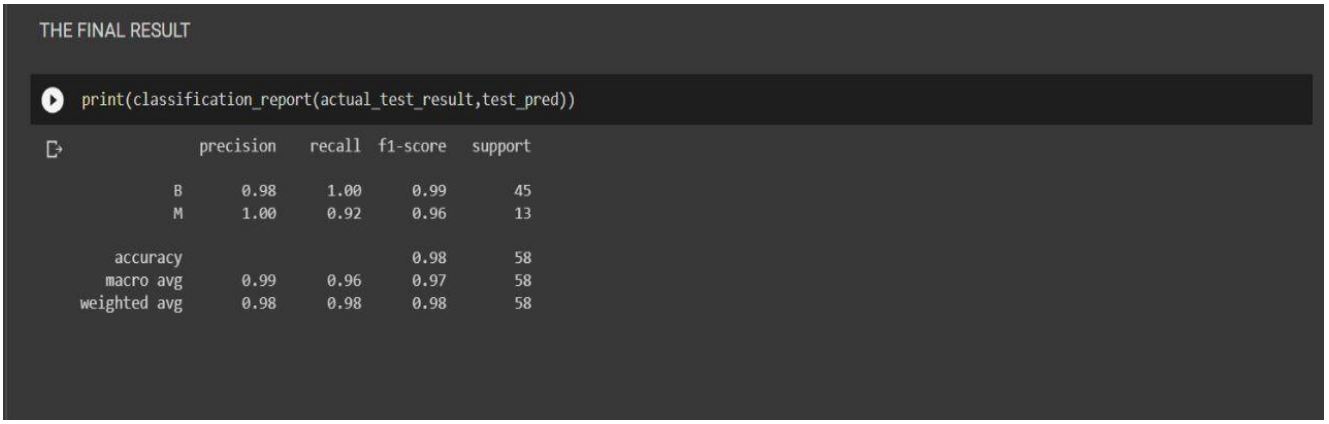
```
[ ] print(classification_report(actual_test_result,test_pred))
```

```
                precision    recall  f1-score   support

             B       0.98      1.00      0.99        45
             M       1.00      0.92      0.96        13

      accuracy                           0.98        58
     macro avg       0.99      0.96      0.97        58
  weighted avg       0.98      0.98      0.98        58
```

RESULT ANALYSIS



Accuracy: The most important metric for evaluating the performance of a machine learning model for cancer detection is accuracy. Accuracy measures the percentage of correct predictions made by the model. A higher accuracy indicates better performance.

Sensitivity and Specificity: Sensitivity measures the proportion of true positives that are correctly identified by the model, while specificity measures the proportion of true negatives that are correctly identified by the model. A good cancer detection model should have high sensitivity and specificity.

Feature Importance: Feature importance measures the contribution of each input feature to the model's predictions. It can be used to identify the most important features for cancer detection and to improve the model's performance by selecting only the most relevant features.

Overall, the result analysis for cancer detection using machine learning involves evaluating the performance of the model using metrics such as accuracy, sensitivity, specificity, ROC curve, and confusion matrix. It also involves analysing the importance of each input feature and identifying ways to improve the model's performance.

CONCLUSION

In this review, we discussed the concepts of ML while we outlined their application in cancer prediction/prognosis. Most of the studies that have been proposed the last years and focus on the development of predictive models using supervised ML methods and classification algorithms aiming to predict valid disease outcomes. Based on the analysis of their results, it is evident that the integration of multidimensional heterogeneous data, combined with the application of different techniques for feature selection and classification can provide promising tools for inference in the cancer domain.

FUTURE SCOPE

Cancer detection using machine learning has a very promising future, with several potential applications and advancements on the horizon. Here are some of the future scope of cancer detection using machine learning:

Personalized Cancer Diagnosis and Treatment: Machine learning algorithms can be trained on large datasets of patient information to develop personalized cancer diagnosis and treatment plans. This can lead to better patient outcomes, reduced healthcare costs, and more efficient use of resources.

Early Cancer Detection: Machine learning algorithms can detect cancer at an early stage, before symptoms appear, which can lead to more effective treatment and higher survival rates.

Integration with Other Technologies: Machine learning algorithms can be integrated with other technologies, such as imaging and genomics, to improve cancer detection and treatment.

Real-time Cancer Detection: Machine learning algorithms can be deployed in real-time, enabling healthcare professionals to detect cancer as it develops, monitor disease progression, and adjust treatment plans accordingly.

Improved Accuracy and Speed: Machine learning algorithms can analyze large amounts of data quickly and accurately, leading to more precise cancer detection and faster diagnosis and treatment.

Remote Monitoring: Machine learning algorithms can be used to monitor cancer patients remotely, enabling healthcare professionals to detect any changes in disease progression and adjust treatment plans accordingly.

Prevention and Risk Assessment: Machine learning algorithms can be used to identify individuals who are at high risk of developing cancer, enabling healthcare professionals to develop personalized prevention strategies and targeted screening programs.

Overall, the future of cancer detection using machine learning looks bright, with the potential to improve patient outcomes, reduce healthcare costs, and accelerate progress in cancer research.

REFERENCE:

[1] https://colab.research.google.com/drive/1M-LJ5IEXhy4GLOm8m8mPo5DIavy87iGU#scrollTo=iNmO4ht6CJ-A

[2] https://drive.google.com/file/d/1UoBZETXmRvYjLgjng3Xwa95ne6F_geUq/view?usp=share_link

[3] http://refhub.elsevier.com/S2001-0370(14)00046-4/rf0015

[4] http://refhub.elsevier.com/S2001-0370(14)00046-4/rf0075

[5] http://refhub.elsevier.com/S2001-0370(14)00046-4/rf0195

[6] http://refhub.elsevier.com/S2001-0370(14)00046-4/rf0300

[7] http://refhub.elsevier.com/S2001-0370(14)00046-4/rf0110