A HYBRID ALGORITHM FOR BREAST CANCER DETECTION USING META LEARNING AND ANN

A PROJECT REPORT

for

SOFT COMPUTING (ITE1015)

in

B.Tech – Information Technology and Engineering

by

ARPAN ANAND (19BIT0088)

SHUBHAM KUMAR (19BIT0119)

Under the Guidance of

Dr. AGILANDEESWARI L

Associate Professor, SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

December, 2021

DECLARATION BY THE CANDIDATE

We hereby declare that the project report entitled **"*A HYBRID ALGORITHM FOR BREAST CANCER DETECTION USING META LEARNING AND ANN*"** submitted by us to Vellore Institute of Technology University, Vellore in partial fulfilment of the requirement for the award of the course **Soft Computing (ITE1015)** is a record of Bonafede project work carried out by us under the guidance of **Dr. Agilandeeswari L.** We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

Place: Vellore

Signature

Shubham Kumar

Arpan Anand.

Date: 28.11.2021

School of Information Technology & Engineering [SITE]

<u>CERTIFICATE</u>

This is to certify that the project report entitled "***A HYBRID ALGORITHM FOR BREAST CANCER DETECTION USING META LEARNING AND ANN***" submitted by **shubham kumar (19BIT0119) and Arpan Anand (19BIT0088)** to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Soft Computing (ITE1015)** is a record of Bonafede work carried out by them under my guidance.

Dr. Agilandeeswari L

GUIDE

Associate Professor, SITE

Abstract

The cases of Breast Cancer are increasing day by day leading to more cancer-related deaths among women which can be prevented if diagnosed early. In this report, we have analysed the various methods used by the researchers, their advantages and limitations and we have tried to use, various supervised and unsupervised learning models like Random Forest, KNN, SVM, Logistic Regression, AdaBoost, and Perceptron and feature selection (Tree-Based Feature Importance) to try to create a model to increase the accuracy of the model to predict breast cancer. The dataset used for this is Wisconsin Breast Cancer Dataset which has 569 instances of breast cancer patients. Metrics like Accuracy, Precision, Recall, F1-Score, ROC-curve is used to judge the performance of the model

Keywords - Breast Cancer, Stacking Classifier, SVM, Random Forest, Logistics Regression, adam, XGBoost, KNN, AdaBoost, Meta Learning, ANN,C4.5, CART

Introduction

Machine Learning is a collection of techniques for efficient and automated discovery of previously unknown patterns in large databases. Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends in the cases of breast cancer helping in diagnosing breast cancer in early stages.

Various supervised and unsupervised learning models like Random Forest, KNN, SVM, Logistic Regression, AdaBoost, and Perceptron. These models are used for classifying the Wisconsin Breast Cancer Dataset (WBCD) from UCI Machine learning depository. In this report we will try to use feature selection to increase the accuracy using meta-learning.

Motivation:

Breast cancer is the most common cancer among women worldwide, claiming the lives of hundreds of thousands of women each year and affecting countries at all levels. It impacts 2.1 million women each year, and also causes the greatest number of cancer-related deaths among women. If it is diagnosed in early stages, the chances of survival are higher. The detection of the pattern of symptoms using machine learning is a very important technique to correctly understand hidden patterns.

OBJECTIVES

The primary objectives to be achieved via this project are:

- Achieve a higher level of accuracy in prediction.

- Create our own hybrid algorithm by combining meta learning and neural network.
- Comparison of various models with our own hybrid model.
- Create an interface where numeric data is entered, and the algorithm predicts for breast cancer.

LITERATURE SURVEY:

PAPER1:

Breast cancer prediction via machine learning.[1]

Authors & Year	Technique used	Advantage	Issue	Metrics used
M. S. Yarabarla, L. K. Ravi, and A. Sivasangari	Computer Aided Diagnoses System (CAD) for breast cancer prediction.	Both classification and regression method random forest algorithm provided the highest accuracy,it is a mixture of many train models that provides the predictions about different training classifiers. Hybrid method was developed to performed the accurate computation on UCI online dataset that provide the mode accuracy results.	Expected probabilities of occurrence and non-occurrence are calculated through K fold cross validation. Which is more expensive task. Data pre-processing Stage took too much time, because it was converted raw data into the valuable form, also the number of patient that are already mentioned in a list were not be considered.	Precision Recall F1-Score Support

PAPER 2:

On the scalability of machine learning algorithms for breast cancer prediction in big data context. [2]

Authors	Technique used	Advantage	Issue	Metrics used
S. Alghunaim and H. H. Al-Baity	Performance Comparison of classification algorithm on Weka and Spark	Support vector machine based On parallel computation, have strength to analyze the multiple data at same time, it provide the highest accuracy rate on two different tool Weka and spark Error rate and	Gene Expression data collection is one of the difficult task. To achieve the good result of accuracy, precision and sensitivity of data, large number of samples was needed for computations.	Accuracy Error Rate

		computation time of SVM is lower than the decision tree and random forest		

PAPER3:

A comparative analysis of nonlinear machine learning algorithms for breast cancer detection.[3]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
A. A. Bataineh	Non Linear machine learning algorithm comparison	When dataset are linearly separable it provides good accuracy level MLP is consists of different layers each layer perform one single task separately, so the computation of this algorithm was faster enough.	User was responsible to set the hidden layers for MLP algorithm. Setting some value sometimes provided under fitting and sometimes over fitting results.Without 10 fold cross validation, it is impossible to predict the accuracy rate from train data models.	Accuracy Precision Recall

PAPER 4:

Comparison of machine learning methods for breast cancer diagnosis.[4]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
E. A. Bayrak, P. Kirci, and T. Ensari	Comparison of SVM and ANN for breast cancer prediction.	After comparison most suitable technique for the prediction of breast cancer was found SVM because the classes are separated through hyper line that provide the more accuracy result than ANN.	Expected probabilities of occurrence and non-occurrence are calculated through K fold cross validation. Which is more expensive task.	Accuracy Precision Recall ROC Area

PAPER 5:

Automated breast cancer diagnosis based on machine learning algorithms.[5]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**

H. Dhahri, E. Al Maghayreh, A. Mahmood, W. Elkilani, and M. Faisal Nagi	Optimization of algorithms through Genetics programmmg technique.	Comparative analysis of different machine algorithm was performed after, selecting some feature through polynomial features operator. Extra tree classifier obtained the highest accuracy than other algorithms.	It took too much time during the evaluation process and model training.GP algorithm was designed to solve the hyper parameter problem but this algorithm process time was too slow.	Accuracy Precision Recall ROC Area F1-Score

PAPER 6:

Breast cancer prediction and detection using data mining classification algorithms: A comparative study.[6]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
M. K. Keles	Comparative analysis of Data Mining Classifier for cancer prediction and detection.	Random forest provided the highest accuracy during evaluation, this algorithm require less efforts. Random forest algorithm do not require the standardization and normalization Of data also can handle nonlinear data more efficiently.	Separate model was designed to check that whether there is a tumor or not. This model took too much processing time. K fold cross validation tech nique are applied for n number of iteration, just to get the desire result, each iteration took too much time.	Accuracy

PAPER 7:

Breast cancer risk prediction using data mining classification techniques.[7]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
K. Williams, P. A. Idowu, J. A. Balogun, and A. I. Oluwaranti	Data Mining classification techniques for risk prediction of breast cancer	Naive Bayes provided the less error rate while computations. Number of the attribute was increases while increases the sample size of	Expression rule was designed to show the best attributes for breast cancer prediction but the evaluation process was too complicated	TP rate FP rate Precision ROC Area

		data that provided the good accuracy results.		

PAPER 8:

Classification algorithm-based analysis of breast cancer data.[8]

Authors	Technique used	Advantage	Issue	Metrics used
B. Padmapriya and T. Velmurugan	J48 CART Decision Tree	Comparison of each classification algorithm was done thmugh the evaluation of weighted average values. CART algorithm provided the better accuracy for prognoses of breast cancer with minimum time.	Model was design to comparatively analyzed the data mining decision algorithm J48,CART,ADtree.Evaluation phase took too much time.	Specificity Sensitivity Accuracy Precision Recall F-Measure

PAPER 9:

Using machine learning algorithms for breast cancer risk prediction and diagnosis.[9]

Authors	Technique used	Advantage	Issue	Metrics used
H. Asri, H. Mousannif, H. Al Moatassime, and T. Noel	C4.5 SVM NB KNN	Evaluation of each classifier through confusion matrix shows that accuracy rate of SVM is higher than the other SVM provided the less error rate for prognoses of breast cancer	the process time of SVM was higher than the KNN algorithm but KNN was a lazy learner method that had not provided the good accuracy result	TP FP Precision Recall F-Measure

PAPER 10:

A study on prediction of breast cancer recurrence using data mining techniques.[10]

Authors	Technique used	Advantage	Issue	Metrics used

U. Ojha and S. Goel	K-Means KNN C5.0 SVM Naïve Bayes	Classification algorithm C45 and SVM provided the better result than the other algorithm. EM was also founded the most appropriate clustering algorithm for breast cancer.	Finding the effect algorithm that predict the accruing and recurring of diseases is one of the most difficult task.	accuracy, sensitivity specificity

PAPER 11:

Cancer disease prediction using naive Bayes ,K-nearest neighbor and J48 algorithm[11]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
S. K. Maliha, R. R. Ema, S. K. Ghosh, H. Ahmed, M. R. J. Mollick, and T. Islam	Naive Bayes, KNN J48.	Most suitable technique for the prediction of cancer dataset Classified the data according to the similarity of each instances Provide the good accuracy for both training data and testing data	Testing phase is slow and also take too much time Difficult to choose require K value. To predict about the new data K-nearest only find the nearest neighbor fmm training data.	Accuracy Error Rate Sensitivity Specificity Precision F-score

PAPER 12:

Breast cancer detection in the IOT health environment using modified recursive feature selection.[12]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
M. H. Memon, J. P. Li, A. U. Haq, M. H. Memon, and W. Zhou	SVM RFE	SVM linear kernel provide the highest accuracy while the selection of appropriate features for breast cancer prediction. Predicted model and features selection technique was designed for computation of large dataset, that provide the good accuracy.	Computation time was increases while the extraction of irrelevant features. Error rate Of SVM linear kernel was higher than others, while the computation process was slower.	MCC Sensitivity Specificity F-score Accuracy

PAPER 13:

Machine learning classification techniques for breast cancer diagnosis.[13]

Authors	Technique used	Advantage	Issue	Metrics used
D. A. Omondiagbe, S. Veeramani, and A. S. Sidhu	SVM ANN Naïve Bayes CFS RFE PCA LDA	Classification model was built through training dataset, this phase took consume too much time during preprocessing. Features selection and extraction help to identify the presences of tumor also improve the classification of benign and malignant patients. Featuæs extraction decrease the data storages issue efficiently.	R programming language is used for the implementation, this language consist of lots of packages, processing is lesser than the other languages. Evaluation phase took too much time because of CFS, LDA, PCA method	Accuracy Sensitivity Specificity Precision F-score Kappa Statistics

PAPER 14:

Using machine learning algorithms for breast cancer risk prediction and diagnosis.[14]

Authors	Technique used	Advantage	Issue	Metrics used
A. Bharat, N. Pooja, and R. A. Reddy	KNN Naïve Bayes CART SVM	ROC curve provide the good evaluation of each algorithm. Prediction of correctively classified instances rate higher through SVM algorithm. Also this algorithm provided lower error rate value.	Processing time of SVM was 0.007 while KNN was 0.01 s.Model was designed to train data for the evaluation of correctly and in correctively classify the instances that was difficult and complex task.	Accuracy

PAPER 15:

Applying best machine learning algorithms for breast cancer prediction and classification.[15]

Authors	Technique used	Advantage	Issue	Metrics used

Y. Khourdifi and M. Bahaj	K-Nearest Neighbors, Support Vector Machines, Naive Bayes, Random Forest	Predictive model was designed, SVM provided the 99.7% accuracy for benign class and 94.6% for malignant class. Turnaround time and error rate of SVM is lesser than other algorithm.	Good and appropriate selection of method is important for evaluation of machine learning algorithm Confusion matrix was design for expected class result. matrix correctively predict the instances but with prediction time was maximum	Accuracy Precision Recall ROC Area

PAPER 16:

Classification based on clustering model for predicting main outcomes of breast cancer using hyper-parameters optimization.[16]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
A. A. Said, L. A. Abd-Elmegid, S. Kholeif, and A. Abdelsamie	Hyper parameter Optimization for Breast Cancer Prediction.	Hyper parameter through clustering method provided the highest accuracy. Hyper parameter handled both categoriCal and continues type of data more effectively.	selected features also provided some redundant data. BCOAP model consists of too many phases each phase took lost if time for the evaluation of breast cancer data	Efficiency Accuracy

PAPER 17:

Artificial neural network for prediction of breast cancer.[17]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
P. Singhal and S. Pareek	Artificial Neural Network for breast cancer.	Multi-layered Neural network created weight arbitrary that provided the Mean Square Error whose rate is too less. Feed forward algorithm help to reduce the error through weight modification.	Requile high processing and time for large number of data, that affect the overall accuracy of data To achieve the good accuracy, precision and sensitivity of data, large number of sample are needed for computations.	Precision Accuracy

PAPER 18:

A comparison of open source data mining tools for breast cancer classification.[18]

Authors	Technique used	Advantage	Issue	Metrics used
A. A. Ibrahim, A. I. Hashad, and N. E. M. Shawky	Comparison of data mining for breast cancer classification.	Single classification provided the highest accuracy than the fusion classificaLion. WPBC, WBC, LBCD Dataset was provided the better level accuracy during the evaluation of different algorithm when the confusion Metrix was design.	Weka tool provided the best accuracy for WPBC and WBC dataset but the accuracy level was not good for LBCD dataset.	Accuracy

PAPER 19:

A Study of the Suitability of Autoencoders for Pre-processing Data in Breast Cancer Experimentation.[19]

Authors	Technique used	Advantage	Issue	Metrics used
Macías-García Laura, Luna-Romera José María, García-Gutiérrez Jorge, Martínez-Ballesteros María, Riquelme-Santos José C. and González-Cámpora Ricardo 2017	Autoencoders to improve the quality of the data.	Autoencoders could statistically be a valuable tool to reduce noise in data (related to breast cancer but potentially to any other biomedical research area) using hidden relationships between biomarkers.	Autoencoders which are used for pre-processing have the risk of overfitting the data, especially when the parameterization was carried out with all the dataset.	p-values, z-value, Holm's α

PAPER 20:

Machine learning applications in cancer prognosis and prediction.[20]

Authors	Technique used	Advantage	Issue	Metrics used
		Several ML techniques were employed as an aim to find the most optimal one.	One of the most common limitations noted in the studies surveyed in this	Accuracy

Konstantina Kourou, Themis P, Exarchos, Konstantinos P, Exarchos, Michalis V, Karamouzis, Dimitrios I and Fotiadis 2015	ANN SVM	It has been found 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.	review is the small amount of data samples. A basic requirement when using classification schemes for modelling a disease is the size of the training datasets that needs to be sufficiently large.	

PAPER 21:

Recent advancement in cancer detection using machine learning: Systematic survey of decades: Comparisons and Challenges.[21]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
TanzilaSaba 2020	CNN	This paper proposed a CNN based method for the detection of breast carcinoma utilizing an unmonitored pathway network of deep-faith beliefs accompanied by a backward propagation route. Wisconsin Breast Cancer Dataset employed for experiments and 99.68% accuracy claimed.	The algorithm for deep learning has improved the precision of the breast cancer diagnosis to 97%, and but processing time exceeded from 30 to 40 s.	Accuracy Specificity Sensitivity F-Score MCC Precision

PAPER 22:

Breast Cancer Prediction using Feature Selection and Ensemble Voting.[22]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Quang H. Nguyen, Trang T.T. Do, Yijing Wang,		This paper compares various Algorithms based on different metrics which in	The number of baseline models has to be in odd numbers and greater than one.	Precision, recall, ROC-AUC, F1-measure,

Sin Swee Heng, Kelly Chen, Wei Hao Max Ang, Conceicao Edwin Philip 2019	Random Forest, KNN, SVM, Logistic Regression, AdaBoost, Perceptron	turn help to identify the best algorithms to be used for this dataset.	XGBoost, Adaboost, SGD and SVM are considered black box models subject to acceptance among industrial practices and regulations.	computational time

PAPER 23:

Analysis of feature selection with classification: Breast cancer datasets.[23]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Lavanya Doddipalli, K. Usha Rani 2011	Decision Trees	In this paper experimental results show that Feature Selection, a Preprocessing technique greatly enhances the accuracy of classification.	-------	Reduced No. of Attributes, Accuracy, Time, Tree size

PAPER 24:

Support vector machines combined with feature selection for breast cancer diagnosis.[24]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Mehmet Fatih Akay 2009	SVM-based model using grid search, F-score	This paper shows that the proposed method yields the highest classification accuracies for a subset that contained five features.	-------	Accuracy, sensitivity, specificity, positive predictive value, negative predictive value, ROC curves and confusion matrices

PAPER 25:

A new classifier for breast cancer detection based on Naïve Bayesian.[25]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**

Murat Karabatak 2015	NB classifier	The applied weighted NB obtained 99.11% sensitivity, 98.25% specificity and 98.54% the accuracy values respectively	Algorithm uses a grid search mechanism to find the optimum weight values. This search was computationally expensive and the initialization of the weights vector is crucial and application dependent	sensitivity, specificity accuracy.

PAPER 26:

Investigating the effect of Correlation based Feature Selection on breast cancer diagnosis using Artificial Neural Network and Support Vector Machines.[26]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Reem Alyami, Jinan Alhajjaj, Batool Alnajrani, Ilham Elaalami, Abdullah Alqahtani, Nahier Aldhafferi, Taoreed O. Owolabi b, and Sunday O. Olatunji 2017	SVM and ANN combined with feature selection	Based on results achieved, both SVM and ANN are observed and compared by means of classification accuracy. SVM showed better performance results on classifying the samples with 97.1388 % accuracy compared to ANN that achieved 96.7096 % accuracy	The duality feature in SVM limits the user to deal with the data as two classes. Expensive implementation due to its training computation.	accuracy tests

PAPER 27:

Breast Cancer Prediction: A Comparative Study Using Machine Learning Techniques[27]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Md. Milon Islam · Md. Rezwanul Haque · Hasib Iqbal · Md. Munirul Hasan ·	support vector machine, K-nearest neighbors,	The developed model by ANNs is	The lowest accuracy derived from	confusion matrix

Mahmudul Hasan · Muhammad Nomani Kabir 2020	random forests, artifcial neural networks, and logistic regression	more consistent than any other technique stated, and it may be able to bring changes in the feld of prediction of breast cancer.	the RFs and LR is 95.7%.	

PAPER 28:

Predicting breast cancer 5-year survival using machine learning: A systematic review[28]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
iaxin Li,Zijun Zhou,Jianyu Dong,Ying Fu,Yuan Li,Ze Luan,Xin Pen 2021	PROBAST	this is the first systematic review of the application of ML to breast cancer survival prediction, and accurate 5-year survival predictions are very important for further research. .	the information on predictive performance (such as true positive, false positive, true negative, and false negative in the confusion matrix) was insufficient, and most of the studies only described a single dimension of predictive performance.	accuracy

PAPER 29:

Model Selection for Predicting Breast Cancer using Supervised Machine Learning Algorithms.[29]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Ajit Kumar ; Rajkumar Patra; Anupam Ghosh 2020	Logistics Regression, K-Nearest Neighbors, Decision Tree Classifier, Gaussian NB, and Support Vector Machine	the dataset contains 32 features so, dimensional reduction helps in decreasing the multidimensional data into few dimensions. On the whole, the above study proposed that Logistic Regression is efficient for the detection of breast cancer as compared to all the other models while dealing	----------------	accuracy, precision, recall, and f-score

		with the complex dataset.		

PAPER 30:

Efficient Breast Cancer Prediction Using Ensemble Machine Learning Models[30]

Authors	**Technique used**	**Advantage**	**Issue**	**Metrics used**
Naveen R. K. Sharma Anil Ramachandran Nair	decision tree, support vector machine, multilayer perceptron, K-nearest neighbors, logistics regression and random forest	Decision tree and KNN gives 100% accuracy with ensemble technique.	---------------	precision, recall, F1-score,accuracy

Conclusion:

The recent works have used various methods in their papers, such as, KNN, SVM, CNN, and grouped them together to increase the various metrics used for the evaluation of their models. Making the model to predict Breast Cancer involves many stages and each stage has its own importance in defining and enhancement of the model. The various types of modalities used for Feature Selection, Classification used in the paper, strategies to make the model better, their advantages and disadvantages and characteristics were discussed in this report. The problems identified by others were solved by others. Although much research work has been done in this field but the scope of improvement is still there with the rise in technologies. And that what we have tried to achieve in this paper via our proposed hybrid algorithm for predicting breast cancer via creating a combination of various kinds of meta learning models whose output was used as input features for the creation of the ANN model, which in turn gave a better performance than other model used in the papers.

Issues in Existing Systems and Conclusions Drawn:

- Some systems had very high hardware requirements.
- Most papers use K means clustering or CNNs.
- We’ve seen that hybrid approaches work better than normal approaches.
- Which is why we thought of combining various preprocessing algorithms and used a combination of algorithms (hybrid algorithm)
- Most of the papers does not provide a principal component analysis of the model and give various metrics like specificity, efficiency, accuracy and sensitivity.
- Neural network is found to be relatively less accurate, precise, reliable compared to other machine learning models.
- Some models come with much lesser accuracy as compared to others

MODULES AND ITS DESCRIPTION

Dataset

The data used is the Wisconsin Breast Cancer Dataset (WBCD) taken from the UCI machine learning repository. The dataset contained 569 instances taken from needle aspirates from patients' breasts, of which 357 cases were identified as "benign" and the remaining 212 cases were classified as "malignant". Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Dataset: To be read and stored using Pandas (data used is Wisconsin Breast Cancer Dataset (WBCD))

Data Pre-Processing

Data preprocessing involves the transformation of the raw dataset into an understandable format. Preprocessing data is a fundamental stage to improve data efficiency. The data preprocessing methods directly affect the outcomes of any analytic algorithm. So, pre-processing is required to increase efficiency of the model.

In this report we will be focusing on four things:

Name	Description
Checking for Missing Values	This check ensures that the conclusion of the model is not affected by missing values within the dataset.

Checking for Outliers	This check ensures that the conclusion of the model is not affected by outlier values within the dataset.
Checking for Class Imbalance	In the dataset, the ratio between the two classes, Benign (B) = 0 and Malignant (M) = 1, is 63:37, respectively. This shows that a close gap of 0.26 exists, which shows that the dataset is pretty much balanced.
Checking for Normalization	All variables should have the same scale for fair comparison between them. There is evident difference in scale for each variable. Therefore, feature scaling is needed to ensure fair treatment.

Feature Selection

Feature Selection is extremely important in any model as it indicates which variables are most efficient and effective for a system.

Types of Feature Selection Methods:

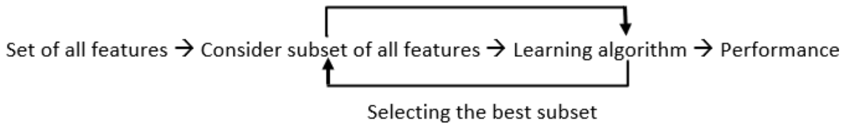
Method Name	Description	Advantages
Filter	Filter methods select features from a dataset independently for any machine learning algorithm. These methods rely only on the characteristics of these variables, so features are filtered out of the data before learning begins. These methods are powerful and simple and help to quickly remove features	• Selected features can be used in any machine learning algorithm, • They’re computationally inexpensive
Wrapper	Wrapper methods work by evaluating a subset of features using a machine learning algorithm that employs a search strategy to look through the space of possible feature subsets, evaluating each subset based on the quality of the performance of a given algorithm.	• It detects the interaction between variables • It finds the optimal feature subset for the desired machine learning algorithm
Embedded	In embedded methods, the feature selection algorithm is blended as part of the learning algorithm in other words, they perform feature selection during the model training.	• They take into consideration the interaction of features like wrapper methods do. • They are faster like filter methods. • They are more accurate than filter methods. • They find the feature subset for the algorithm being trained. • They are much less prone to overfitting.

It can be seen from above that the embedded methods are preferable for feature selection over filter and wrapper methods.

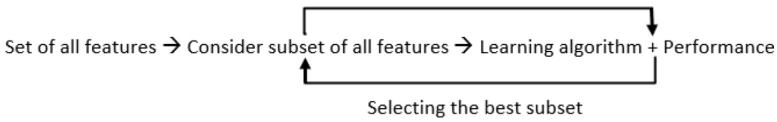
Pictorial Representation of General Filter Feature selection method.

Set of all features → Selecting the best subset → Learning algorithm → Performance

Pictorial Representation of General Wrapper Feature selection method.



Pictorial Representation of General Embedded Feature selection method.



Types of Embedded Feature Selection Methods:

Method Name	Description
Regularization	It adds a penalty to different parameters of the machine learning model to avoid over-fitting of the model. The penalty is applied over the coefficients, thus bringing down some coefficients to zero. The features having zero coefficient can be removed from the dataset.
Tree-based Feature Importance	It tells us which variables are more important in making accurate predictions on the target variable/class. It identifies which features are the most used by the machine learning algorithm in order to predict the target.

From the above it can be seen that the tree-based feature importance is much better option for feature selection than Regularization as it utilizes penalties to reduce the features instead of telling which features affect the model the most.

Thus, we will be using **Tree based feature selection and Random Forest Classification** for feature selection in the Wisconsin dataset.

Classification

METHOD	**DESCRIPTION**	**ADVANTAGES**	**DISADVANTAGES**
Knn algorithm	Classification of predictors according to cluster of similar behaviour. This is a form of optimization that seek to find the nearest point to a target variable point.	• No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. • easy to implement	• It does not work well with large dataset. • It does not work well with high dimensions. • It needs feature scaling

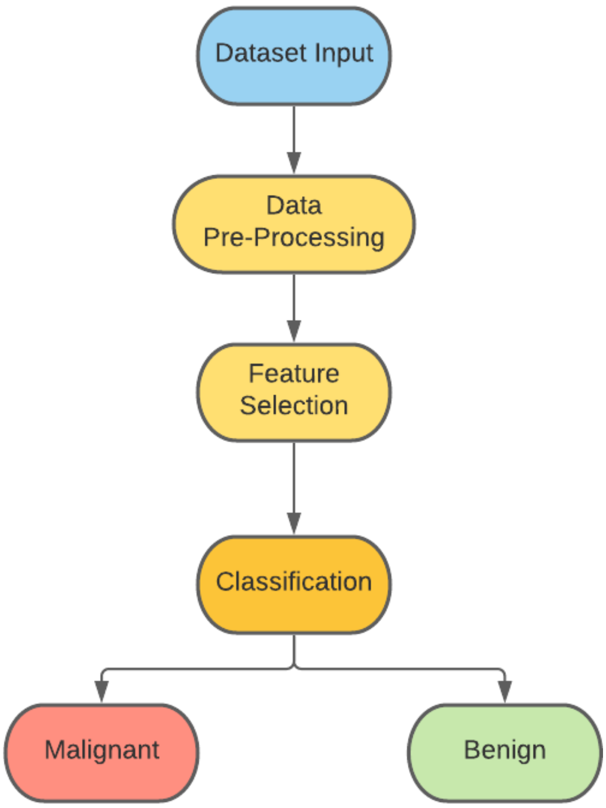
Support Vector Machines	Support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis	• SVM works relatively well when there is a clear margin of separation between classes. • SVM is more effective in high dimensional spaces. • SVM is effective in cases where the number of dimensions is greater than the number of samples. • SVM is relatively memory efficient	• SVM algorithm is not suitable for large data sets. • SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. • In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform
Logistic Regression	Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.	• Logistic Regression performs well when the dataset is linearly separable. • Logistic regression is less prone to over-fitting but it can overfit in high dimensional datasets. You should consider Regularization (L1 and L2) techniques to avoid over-fitting in these scenarios. • Logistic Regression not only gives a measure of how relevant a predictor (coefficient size) is, but also its direction of association (positive or negative).	• Main limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. In the real world, the data is rarely linearly separable. Most of the time data would be a jumbled mess. • If the number of observations are lesser than the number of features, Logistic Regression should not be used, otherwise it may lead to overfit.
Stochastic Gradient Descent	Gradient descent is a method of	• It is easier to fit in the memory due	• Due to frequent updates, the steps

	optimization and stochastic gradient descent (SGD) is an incremental gradient descent for finding the minimum of function. Stochastic is an approximation of gradient descent optimization.	to a single training example being processed by the network. • It is computationally fast as only one sample is processed at a time. • For larger datasets, it can converge faster as it causes updates to the parameters more frequently.	taken towards the minima are very noisy. This can often lean the gradient descent into other directions. • Also, due to noisy steps, it may take longer to achieve convergence to the minima of the loss function. • Frequent updates are computationally expensive because of using all resources for processing one training sample at a time.
Perceptron	Perceptron is a neural network that decides if an input belongs to a specific class, based on weighted feature vector.	• Single Layer Perceptron is quite easy to set up and train. • The neural network model can be explicitly linked to statistical models which means the model can be used to share covariance Gaussian density function. • The SLP outputs a function which is a sigmoid and that sigmoid function can easily be linked to posterior probabilities.	• This neural network can represent only a limited set of functions. • The decision boundaries that are the threshold boundaries are only allowed to be hyperplanes. • This model only works for the linearly separable data.
AdaBoost	Boosting is an ensemble method that start on a base classifier from the training dataset. AdaBoost is a boosting ensemble method which is building on up	• It is very fast, • It is easy to use, • It is easy to program, • It can be combined with any other machine learning algorithm without the requirement of	• It is potentially vulnerable to noise due to its own empirical evidence. • If weak classifier underperform, they can make the whole model underperform, • Adaboost is highly susceptible to

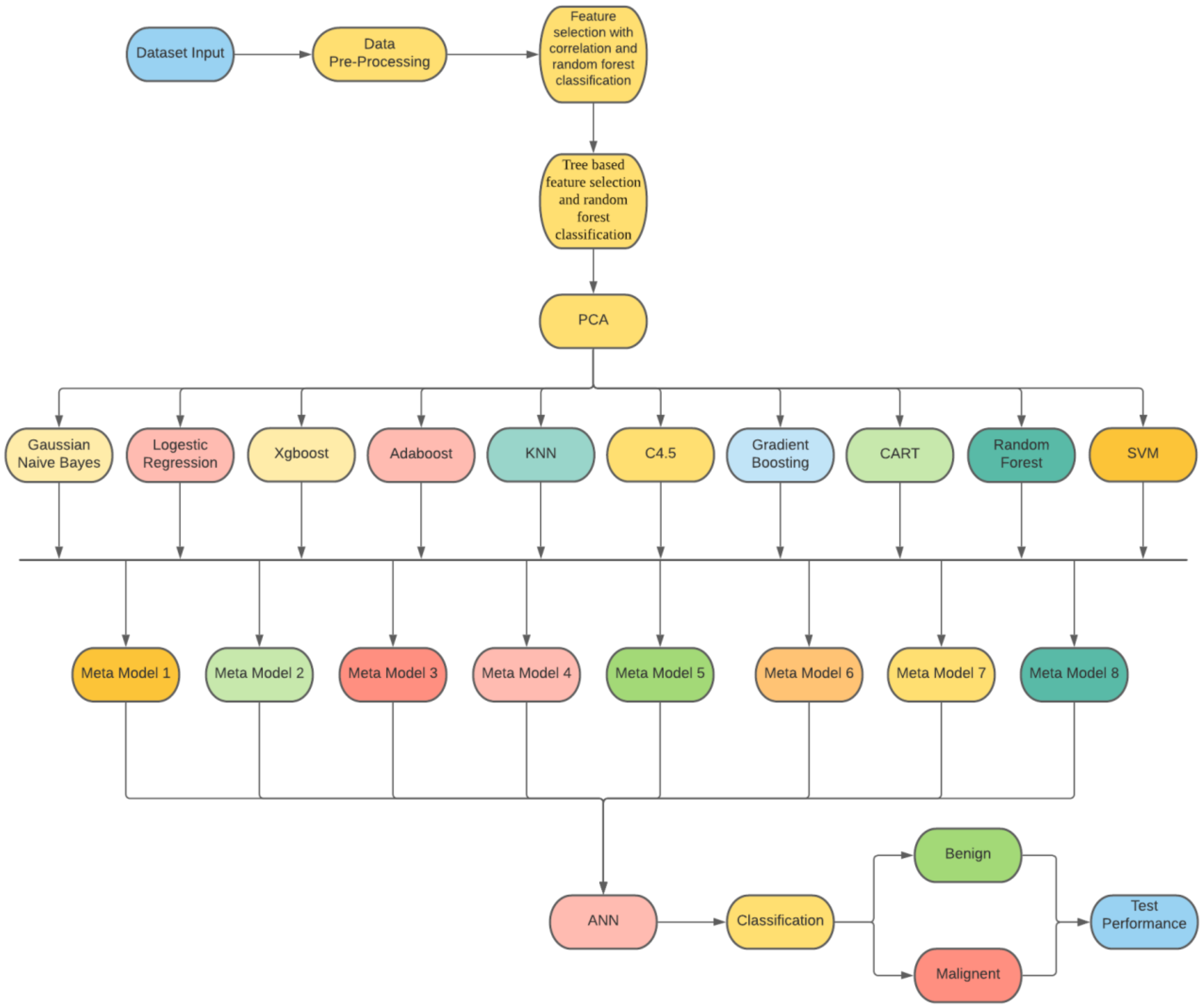
	weighted classification.	fine-tuning parameters. • It can be used in problems which are not in the form of binary classification	outlier. Thus, not useful in scenarios where outliers are expected to happen.
XGBoost	XGBoost is a type of gradient tree boosting which allows for regularization, in order to avoid overfitting.	• It is Highly Flexible • It uses the power of parallel processing • It is faster than Gradient Boosting • It supports regularization	• Difficult interpretation , visualization tough • Overfitting possible if parameters not tuned properly. • Harder to tune as there are too many hyperparameters.

It is understood from above analysis that every learning model has its own advantages and disadvantages which help them to be better or worse than the other learning models. We cannot choose any one learning model over other. So, we will be trying to make a meta learning model which will learn from learning, which will in turn help to better predict the breast cancer. It will do so by training a model over previously trained model.

General Architecture



Proposed architecture diagram:



Novelty

- We have tried to developed our own hybrid approach which has not been used till date in any of the paper or anywhere as far our knowledge.
- The Accuracy which we have achieved is one of the highest among all the papers which we have reviewed.
- We have used various matrices for evaluation such as accuracy, precision, recall, F1 score and confusion matrix.

Metrics used

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$Accuracy = \left(\frac{(TP + TN)}{TP + FP + TN + FN}\right)$$

Precision attempts to answer the question what proportion of positive identifications was actually correctly:

$$TN \,/\, (TN + FP)$$

Recall attempts to answer the following question about what proportion of actual positives was identified correctly.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 Score is the 2*((precision*recall)/(precision+recall))

Pseudocode

1. Take data from the dataset.
2. Pre-process it for any irregularities like outliers/null value etc.
3. Use correlation to select features by reducing the number of highly correlated attributes.
4. Use Tree based feature selection and random forest classification to determine the important attributes.
5. Use PCA to reduce dimensionality
6. Hyper Tune the various algorithms present in the literature review.
7. Compare various algorithms present in the literature review.
8. Select the best algorithms suited for this model.
9. Create various Meta Learning Models by combining various Linear, Non-Linear algorithms to create the meta models.
10. Use their output as the input for ANN Model.
11. Use the ANN model to predict the **Breast Cancer.**
12. The data can be passed through an android app to predict the cancer.

Interface

Flask based interface that allows numeric content to be passed to the model to predict the cancer.

Importing various libraries and data set description-

In [ ]:

```
###Loading the packages.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [ ]:

```
plt.style.use("seaborn")
```

In [ ]:

```
# Loading the data
df = pd.read_csv('data.csv')
df.head()
```

Out[ ]:

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

5 rows × 33 columns

```
print("\nShape = ",df.shape)
```

```
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')

Shape =  (569, 33)
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [ ]:

```
df.describe().T
```

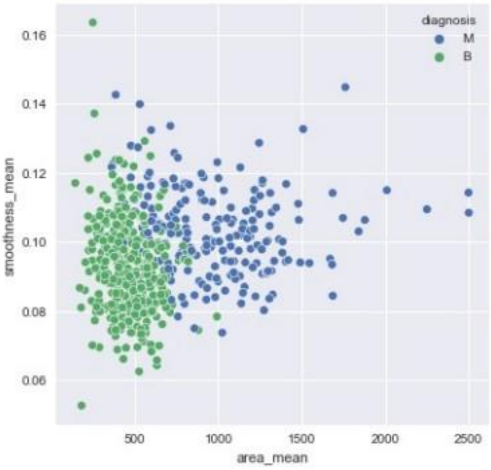
Out[ ]:

	count	mean	std	min	25%	50%	75%	max
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.000000	906024.000000	8.813129e+06	9.113205e+08
radius_mean	569.0	1.412729e+01	3.524049e+00	6.981000	11.700000	13.370000	1.578000e+01	2.811000e+01
texture_mean	569.0	1.928965e+01	4.301036e+00	9.710000	16.170000	18.840000	2.180000e+01	3.928000e+01
perimeter_mean	569.0	9.196903e+01	2.429898e+01	43.790000	75.170000	86.240000	1.041000e+02	1.885000e+02
area_mean	569.0	6.548891e+02	3.519141e+02	143.500000	420.300000	551.100000	7.827000e+02	2.501000e+03
smoothness_mean	569.0	9.636028e-02	1.406413e-02	0.052630	0.086370	0.095870	1.053000e-01	1.634000e-01
compactness_mean	569.0	1.043410e-01	5.281276e-02	0.019380	0.064920	0.092630	1.304000e-01	3.454000e-01
concavity_mean	569.0	8.879932e-02	7.971981e-02	0.000000	0.029560	0.061540	1.307000e-01	4.268000e-01
concave points_mean	569.0	4.891915e-02	3.880284e-02	0.000000	0.020310	0.033500	7.400000e-02	2.012000e-01
symmetry_mean	569.0	1.811619e-01	2.741428e-02	0.106000	0.161900	0.179200	1.957000e-01	3.040000e-01
fractal_dimension_mean	569.0	6.279761e-02	7.060363e-03	0.049960	0.057700	0.061540	6.612000e-02	9.744000e-02
radius_se	569.0	4.051721e-01	2.773127e-01	0.111500	0.232400	0.324200	4.789000e-01	2.873000e+00
texture_se	569.0	1.216853e+00	5.516484e-01	0.360200	0.833900	1.108000	1.474000e+00	4.885000e+00
perimeter_se	569.0	2.866059e+00	2.021855e+00	0.757000	1.606000	2.287000	3.357000e+00	2.198000e+01
area_se	569.0	4.033708e+01	4.549101e+01	6.802000	17.850000	24.530000	4.519000e+01	5.422000e+02
smoothness_se	569.0	7.040979e-03	3.002518e-03	0.001713	0.005169	0.006380	8.146000e-03	3.113000e-02
compactness_se	569.0	2.547814e-02	1.790818e-02	0.002252	0.013080	0.020450	3.245000e-02	1.354000e-01
concavity_se	569.0	3.189372e-02	3.018606e-02	0.000000	0.015090	0.025890	4.205000e-02	3.960000e-01
concave points_se	569.0	1.179614e-02	6.170285e-03	0.000000	0.007638	0.010930	1.471000e-02	5.279000e-02
symmetry_se	569.0	2.054230e-02	8.266372e-03	0.007882	0.015160	0.018730	2.348000e-02	7.895000e-02
fractal_dimension_se	569.0	3.794904e-03	2.646071e-03	0.000895	0.002248	0.003187	4.558000e-03	2.984000e-02
radius_worst	569.0	1.626919e+01	4.833242e+00	7.930000	13.010000	14.970000	1.879000e+01	3.604000e+01
texture_worst	569.0	2.567722e+01	6.146258e+00	12.020000	21.080000	25.410000	2.972000e+01	4.954000e+01
perimeter_worst	569.0	1.072612e+02	3.360254e+01	50.410000	84.110000	97.660000	1.254000e+02	2.512000e+02
area_worst	569.0	8.805831e+02	5.693570e+02	185.200000	515.300000	686.500000	1.084000e+03	4.254000e+03
smoothness_worst	569.0	1.323686e-01	2.283243e-02	0.071170	0.116600	0.131300	1.460000e-01	2.226000e-01
compactness_worst	569.0	2.542650e-01	1.573365e-01	0.027290	0.147200	0.211900	3.391000e-01	1.058000e+00
concavity_worst	569.0	2.721885e-01	2.086243e-01	0.000000	0.114500	0.226700	3.829000e-01	1.252000e+00
concave points_worst	569.0	1.146062e-01	6.573234e-02	0.000000	0.064930	0.099930	1.614000e-01	2.910000e-01

symmetry_worst	569.0	2.900756e-01	6.186747e-02	0.156500	0.250400	0.282200	3.179000e-01	6.638000e-01
fractal_dimension_worst	569.0	8.394582e-02	1.806127e-02	0.055040	0.071460	0.080040	9.208000e-02	2.075000e-01
Unnamed: 32	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

n [ ]:

```
plt.style.use("seaborn")
plt.figure(figsize=(6, 6))
sns.scatterplot(x = df['area_mean'], y = df['smoothness_mean'], hue = df['diagnosis'], data = df)
plt.show()
```



In [ ]:

```
### Seperating the Target feature and other features
y = df.diagnosis                                  # M or B
list = ['Unnamed: 32','id','diagnosis']
x = df.drop(list,axis = 1 )
x.head()
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_m
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1

5 rows × 30 columns

In order to conduct our analysis easily, we have converted the target column as:</br> **Malignant** - 1</br> **Benignant** - 0</br>

In [ ]:

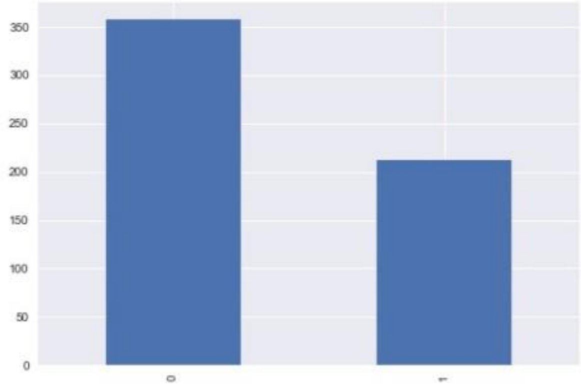
```
y.replace({"M":1,"B":0},inplace=True)
```

In [ ]:

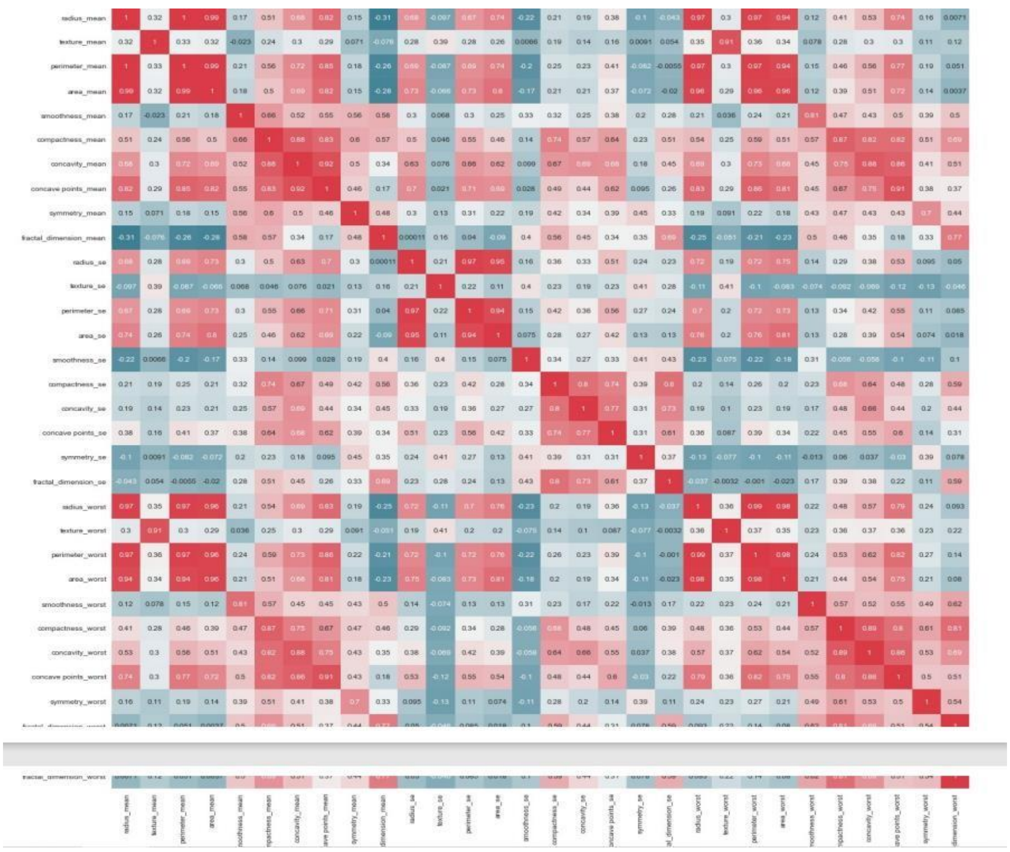
```
df.diagnosis.value_counts().plot(kind='bar')
plt.show()
```

In [ ]:
```
y.replace({"M":1,"B":0},inplace=True)
```

In [ ]:
```
df.diagnosis.value_counts().plot(kind='bar')
plt.show()
```



In [ ]:
```
plt.figure(figsize=(20,20))
sns.heatmap(x.corr(), cmap=sns.diverging_palette(220, 10, as_cmap=True), annot = True,cbar=False)
plt.title("Correlation Map", fontweight = "bold", fontsize=16)
plt.show()
```



Feature selection with correlation and random forest classification

As it can be seen in map heat figure radius_mean, perimeter_mean and area_mean are correlated with each other so we will use only area_mean. If you ask how i choose area_mean as a feature to use, well actually there is no correct answer, I just look at swarm plots and area_mean looks like clear for me but we cannot make exact separation among other correlated features without trying. So lets find other correlated features and look accuracy with random forest classifier. Compactness_mean, concavity_mean and concave points_mean are correlated with each other.Therefore I only choose concavity_mean. Apart from these, radius_se, perimeter_se and area_se are correlated and I only use area_se. radius_worst, perimeter_worst and area_worst are correlated so I use area_worst. Compactness_worst, concavity_worst and concave points_worst so I use concavity_worst. Compactness_se, concavity_se and concave points_se so I use concavity_se. texture_mean and texture_worst are correlated and I use texture_mean. area_worst and area_mean are correlated, I use area_mean.

In [ ]:

```
drop_list1 = ['perimeter_mean','radius_mean','compactness_mean','concave points_mean','radius_se','perimeter_se',
x_1 = x.drop(drop_list1,axis = 1 )
print(x_1.shape)
x_1.head()
```

```
(569, 16)
```

Out[ ]:

	texture_mean	area_mean	smoothness_mean	concavity_mean	symmetry_mean	fractal_dimension_mean	texture_se	area_se	smoothness_se
0	10.38	1001.0	0.11840	0.3001	0.2419	0.07871	0.9053	153.40	0.006399
1	17.77	1326.0	0.08474	0.0869	0.1812	0.05667	0.7339	74.08	0.005225
2	21.25	1203.0	0.10960	0.1974	0.2069	0.05999	0.7869	94.03	0.006150
3	20.38	386.1	0.14250	0.2414	0.2597	0.09744	1.1560	27.23	0.009110
4	14.34	1297.0	0.10030	0.1980	0.1809	0.05883	0.7813	94.44	0.011490

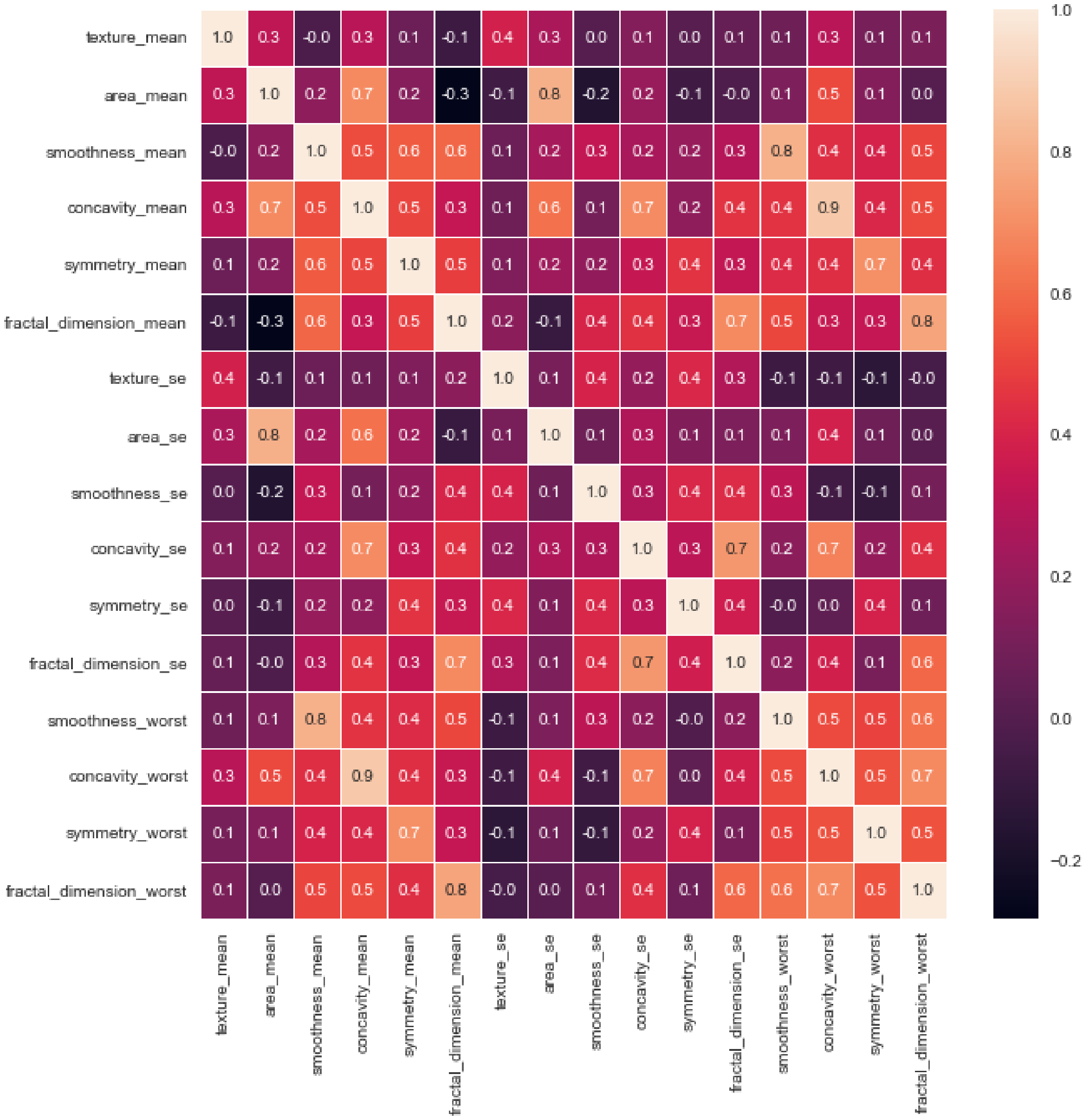
After drop correlated features, as it can be seen in below correlation matrix, there are no more correlated features. Actually, I know and you see there is correlation value 0.9 but lets see together what happen if we do not drop it.

In [ ]:

```
f,ax = plt.subplots(figsize=(10, 10))
sns.heatmap(x_1.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

Out[ ]:

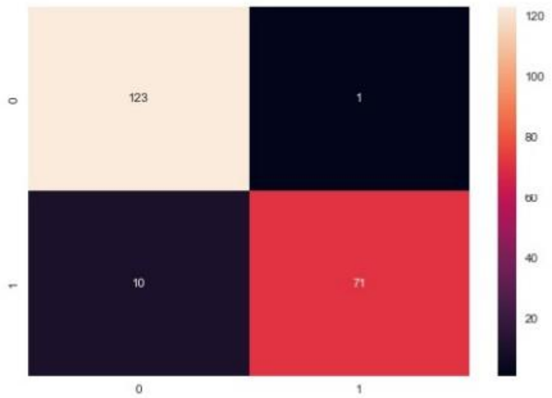
```
<AxesSubplot:>
```


texture_mean
area_mean
smoothness_mean
concavity_mean
symmetry_mean
fractal_dimension_mean
texture_se
area_se
smoothness_se
concavity_se
symmetry_se
fractal_dimension_se
smoothness_worst
concavity_worst
symmetry_worst
fractal_dimension_worst
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
```

In [ ]:
```
# split data train 64 % and test 36 %
x_train, x_test, y_train, y_test = train_test_split(x_1, y, test_size=0.36, random_state=18)

#random forest classifier with n_estimators=10 (default)
clf_rf = RandomForestClassifier(random_state=43)
clr_rf = clf_rf.fit(x_train,y_train)

ac = accuracy_score(y_test,clf_rf.predict(x_test))
print('Accuracy is: ',ac)
cm = confusion_matrix(y_test,clf_rf.predict(x_test))
sns.heatmap(cm,annot=True,fmt="d")
```

```
Accuracy is:  0.9463414634146341
```
Out[ ]:
```
<AxesSubplot:>
```

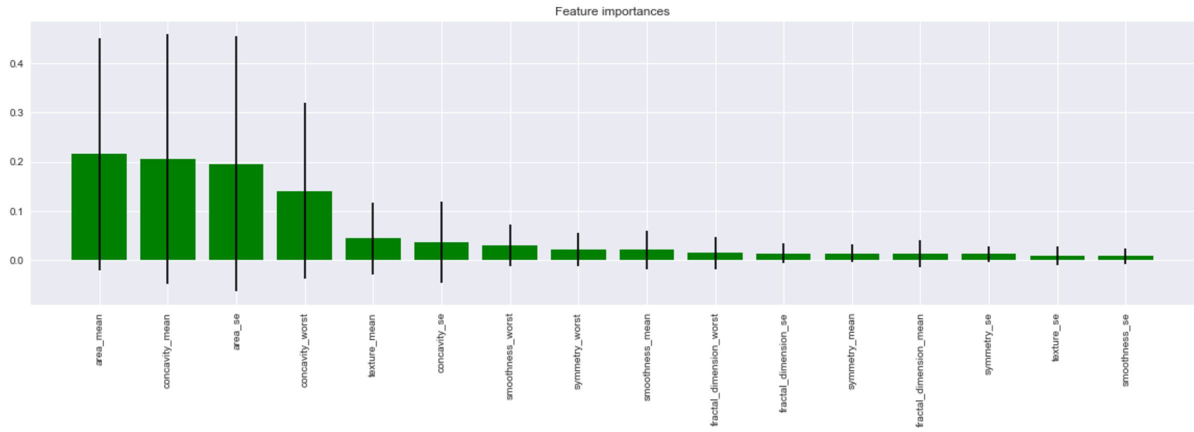


We can see that, we have achieved an accurary of **94.6%** with making a few wrong predictions.

Tree based feature selection and random forest classification

In [ ]:
```
clf_rf_1 = RandomForestClassifier()
clr_rf_1 = clf_rf_1.fit(x_train,y_train)
```

```
importances = clr_rf_1.feature_importances_
std = np.std([tree.feature_importances_ for tree in clf_rf.estimators_], axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
# print("Feature ranking:")

# for f in range(x_train.shape[1]):
#     print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

# Plot the feature importances of the forest

plt.figure(1, figsize=(20, 5))
plt.title("Feature importances")
plt.bar(range(x_train.shape[1]), importances[indices],
       color="g", yerr=std[indices], align="center")
plt.xticks(range(x_train.shape[1]), x_train.columns[indices],rotation=90)
plt.xlim([-1, x_train.shape[1]])

plt.show()
```

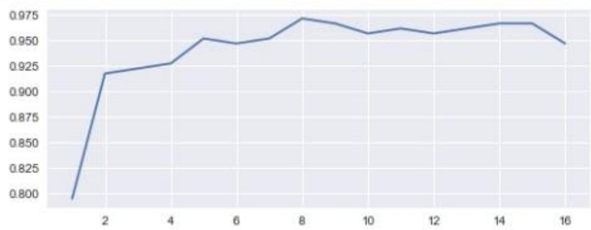


In [ ]:

```python
test_accuracies = []
t = x_train
t_2 = x_test

to_be_removed = []
for f in range(16):
    to_be_removed.append(x_train.columns[indices[f]])

for i in range(16,0,-1):
    clf_rf = RandomForestClassifier(random_state=43)
    clr_rf = clf_rf.fit(t,y_train)
    test_ac = accuracy_score(y_test,clf_rf.predict(t_2))
    test_accuracies.append(test_ac)
    t = t.drop(to_be_removed[i-1],axis=1)
    t_2 = t_2.drop(to_be_removed[i-1],axis=1)
```

In [ ]:

```python
plt.figure(figsize=(8,3))
x_place = [16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]
plt.plot(x_place,test_accuracies)
plt.show()
```



So, we can see that **12 features** give us the **best accuracy = 97.07%**.

Selecting top 12 features required for training the model for feature extraction using **PCA**

In [ ]:

```python
tweleve = indices[:12]
for f in range(len(tweleve)):
    print("%d. feature %d (%f)" % (f + 1, tweleve[f], importances[tweleve[f]]),end=' - ')
    print(x_train.columns[tweleve[f]])
removal_list = ['symmetry_mean','texture_se','symmetry_se','smoothness_se']
x_train_12 = x_train.drop(removal_list,axis=1)
print(x_train_12.shape)
x_train_12.head()
```

```
1. feature 1 (0.215407) - area_mean
2. feature 3 (0.205646) - concavity_mean
3. feature 7 (0.196049) - area_se
4. feature 13 (0.141192) - concavity_worst
5. feature 0 (0.044247) - texture_mean
6. feature 9 (0.036530) - concavity_se
7. feature 12 (0.030493) - smoothness_worst
8. feature 14 (0.022128) - symmetry_worst
9. feature 2 (0.020994) - smoothness_mean
10. feature 15 (0.014807) - fractal_dimension_worst
11. feature 11 (0.014151) - fractal_dimension_se
12. feature 4 (0.013977) - symmetry_mean
(364, 12)
```

Out[ ]:

	texture_mean	area_mean	smoothness_mean	concavity_mean	fractal_dimension_mean	area_se	concavity_se	fractal_dimension_se	smoo
276	14.16	396.6	0.09379	0.001487	0.05821	17.09	0.001487	0.001627	
494	20.54	538.7	0.07335	0.018000	0.05888	26.07	0.013410	0.002701	
239	39.28	920.6	0.09812	0.141700	0.05966	49.00	0.026020	0.002759	
227	15.51	684.5	0.08371	0.065050	0.05907	19.88	0.036440	0.003204	
484	11.28	747.2	0.10430	0.119100	0.06259	13.87	0.033360	0.002256	

In [ ]:

```
x_test_12 =  x_test.drop(removal_list,axis=1)
x_test_12.head()
```

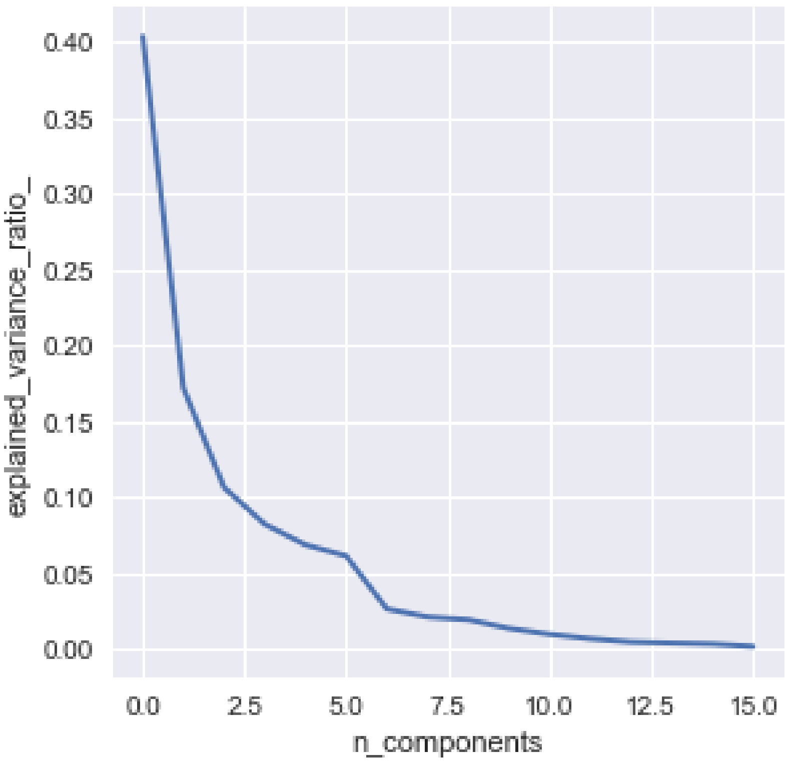
Out[ ]:

	texture_mean	area_mean	smoothness_mean	concavity_mean	fractal_dimension_mean	area_se	concavity_se	fractal_dimension_se	smoo
232	33.81	386.8	0.07780	0.004967	0.05828	15.46	0.003223	0.002534	
490	22.44	466.5	0.08192	0.017140	0.05976	18.04	0.009410	0.002399	
543	28.06	538.4	0.08671	0.029870	0.05781	17.85	0.014980	0.001343	
160	20.18	419.8	0.10890	0.068430	0.06453	38.34	0.041670	0.005061	
8	21.82	519.8	0.12730	0.185900	0.07389	24.32	0.035530	0.003749	

Using PCA for Feature Extraction

In [ ]:

```
#normalization
x_train_N = (x_train-x_train.mean())/(x_train.max()-x_train.min())
x_test_N = (x_test-x_test.mean())/(x_test.max()-x_test.min())

from sklearn.decomposition import PCA
pca = PCA()
pca.fit(x_train_N)

plt.figure(1, figsize=(5, 5))
plt.clf()
plt.axes([.2, .2, .7, .7])
plt.grid(True)
plt.plot(pca.explained_variance_ratio_, linewidth=2)
plt.axis('tight')
plt.xlabel('n_components')
plt.ylabel('explained_variance_ratio_')
plt.show()
```



Using various ML Algorithms

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import GridSearchCV
import xgboost as xgb

import time

from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

In [ ]:
```
acc_train = []
acc_test = []
pres_train = []
pres_test = []
rec_train = []
rec_test = []
f1_train = []
f1_test = []
train_time = []
test_time = []
confusion_matrixs = []
```

In [ ]:
```
def classification_model_report(model,name,n):
    #Fit the model:
    model = model.fit(x_train_12,y_train)

    #Make predictions on training set:
    start_time = time.time()
    pred_train = model.predict(x_train_12)
    end_time = time.time()
    train_time_model = end_time-start_time
    train_time.append(train_time_model)

    start_time = time.time()
    pred_test = model.predict(x_test_12)
    end_time = time.time()
    test_time_model = end_time-start_time
    test_time.append(test_time_model)

    #Print accuracy
    ac_train = accuracy_score(y_train,pred_train)
    ac_test = accuracy_score(y_test,pred_test)
    acc_train.append(ac_train)
    acc_test.append(ac_test)


    #Print precision
    pr_train = precision_score(y_train, pred_train)
    pr_test = precision_score(y_test, pred_test)
    pres_train.append(pr_train)
    pres_test.append(pr_test)
```

```
    #Print recall
    re_train = recall_score(y_train, pred_train)
    re_test = recall_score(y_test, pred_test)
    rec_train.append(re_train)
    rec_test.append(re_test)

    #Print f1_score
    f_train = f1_score(y_train, pred_train)
    f_test = f1_score(y_test, pred_test)
    f1_train.append(f_train)
    f1_test.append(f_test)

    #confusion matrix
    cm = confusion_matrix(y_test,pred_test)
    confusion_matrixs.append(cm)
```

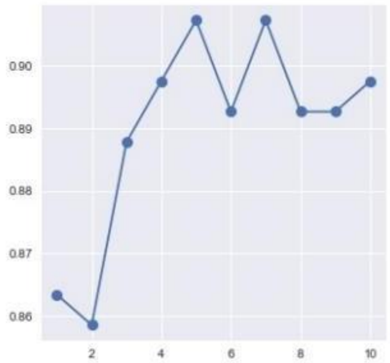
```
if n==1:
  print("|| "+name+" ||\n")
  print("------------------------------------------------------------------------------------------
  print("Trainning\n")
  print("Time: ",train_time_model, end=" || ")
  print("Accuracy: ",round(ac_train,5), end=" || ")
  print("Precision: ",round(pr_train,5), end=" || ")
  print("Recall: ",round(re_train,5), end=" || ")
  print("f1_score: ",round(f_train,5))
  print("\n----------------------------------------------------------------------------------------
  print("Testing\n")
  print("Time: ",test_time_model, end=" || ")
  print("Accuracy: ",round(ac_test,5), end=" || ")
  print("Precision: ",round(pr_test,5), end=" || ")
  print("Recall: ",round(re_test,5), end=" || ")
  print("f1_score: ",round(f_test,5))
  print("\n----------------------------------------------------------------------------------------
```

Hyper tuning various algorithms

KNN

In [ ]:

```
acc_rate=[]

for i in range(1,11):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train_12, y_train)
    pred=knn.predict(x_test_12)
    acc_rate.append(np.mean(pred==y_test))

plt.figure(figsize=(5,5))
plt.grid(True)
plt.plot(range(1,11), acc_rate,marker='o', markersize=9)
plt.show()
```

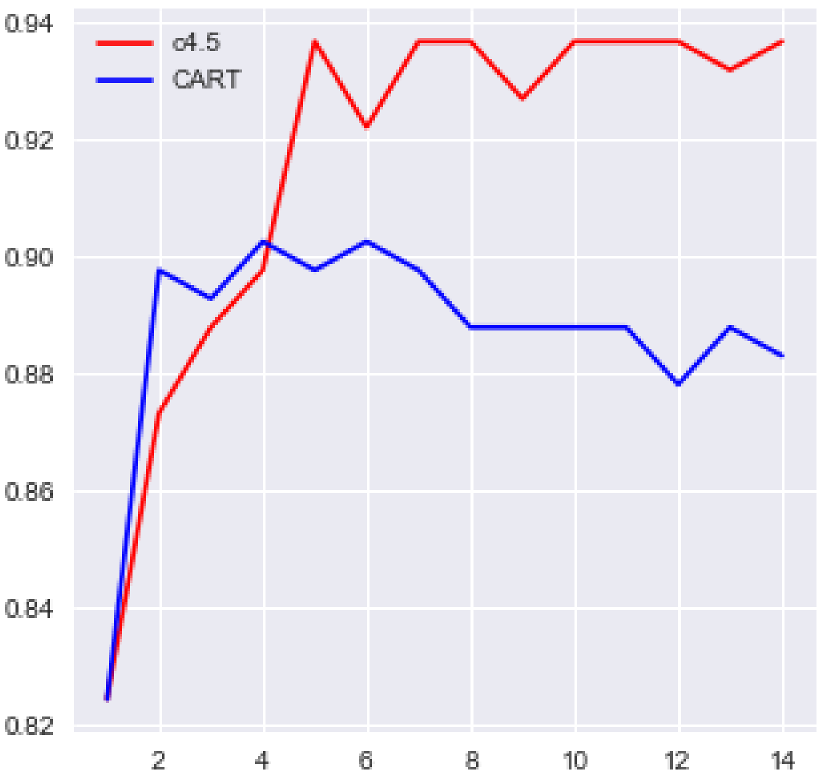


Thus KNN works best at **neighbours=5**.

C4.5 and CART

In [ ]:

```
acc_rate_1=[]
acc_rate_2=[]

for i in range(1,15):
    c4_5 = DecisionTreeClassifier(criterion="entropy", max_depth = i)
    cart = DecisionTreeClassifier(criterion="gini", max_depth = i)
    c4_5.fit(x_train_12, y_train)
    cart.fit(x_train_12, y_train)
    pred1=c4_5.predict(x_test_12)
    pred2=cart.predict(x_test_12)
    acc_rate_1.append(np.mean(pred1==y_test))
    acc_rate_2.append(np.mean(pred2==y_test))

plt.figure(figsize=(5,5))
plt.grid(True)
plt.plot(range(1,15), acc_rate_1,'r')
plt.plot(range(1,15), acc_rate_2,'b')
plt.legend(['c4.5','CART'])
plt.show()
```



Thus, **CART** works best at max_depth = 5 and **C4.5** works best at max_depth=4

SVM

```
In [ ]: svc = SVC()
        parameters = {
            'gamma' : [0.0001, 0.001, 0.01, 0.1],
            'C' : [0.01, 0.05, 0.5, 0.1, 1, 10, 15, 20]
        }

        grid_search = GridSearchCV(svc, parameters)
        grid_search.fit(x_train_12, y_train)
        grid_search.best_params_

Out[ ]: {'C': 15, 'gamma': 0.0001}
```

Gradient Boosting

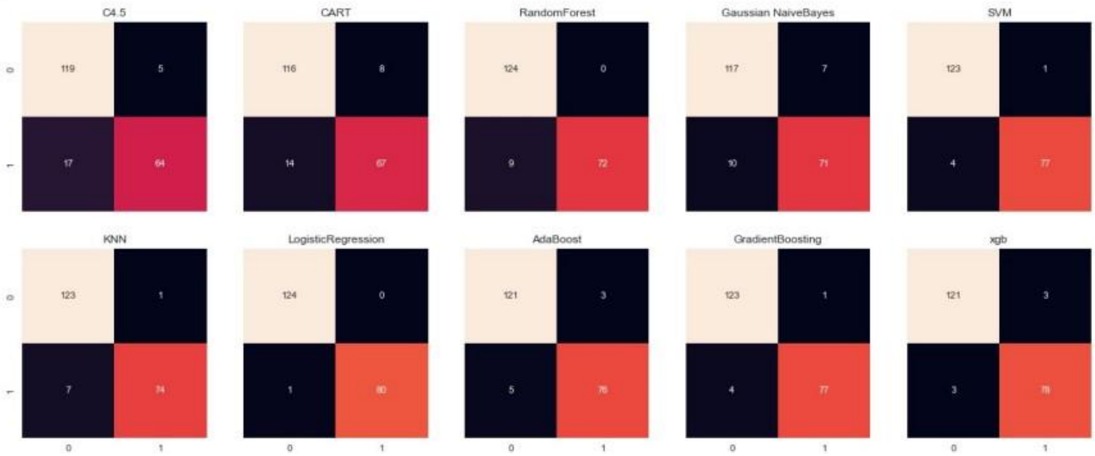
```
In [ ]: gbc = GradientBoostingClassifier()

        parameters = {
            'loss': ['deviance', 'exponential'],
            'learning_rate': [0.001, 0.1, 1, 10],
            'n_estimators': [100, 150, 180, 200]
        }

        grid_search_gbc = GridSearchCV(gbc, parameters, cv = 5, n_jobs = -1, verbose = 1)
        grid_search_gbc.fit(x_train_12, y_train)
        grid_search_gbc.best_params_

        Fitting 5 folds for each of 32 candidates, totalling 160 fits

Out[ ]: {'learning_rate': 1, 'loss': 'exponential', 'n_estimators': 150}
```

Creating ML Models

```
In [ ]:
model_name = ['C4.5','CART','RandomForest','Gaussian NaiveBayes','SVM','KNN','LogisticRegression','AdaBoost','Gra

scaler = StandardScaler()
x_train_12 = scaler.fit_transform(x_train_12)
x_test_12 = scaler.transform(x_test_12)

model1 = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
model2 = DecisionTreeClassifier(criterion="gini", max_depth = 5)
model3 = RandomForestClassifier(n_estimators=60, random_state=0)
model4 = GaussianNB()
model5 = SVC(C = 15, gamma = 0.01)
model6 = KNeighborsClassifier(n_neighbors = 5)
model7 = LogisticRegression()
model8 = AdaBoostClassifier()
model9 = GradientBoostingClassifier(learning_rate = 1, loss = 'exponential', n_estimators = 150)
model10 = xgb.XGBClassifier(random_state=0,booster="gbtree")

models = [model1,model2,model3,model4,model5,model6,model7,model8,model9,model10]
```

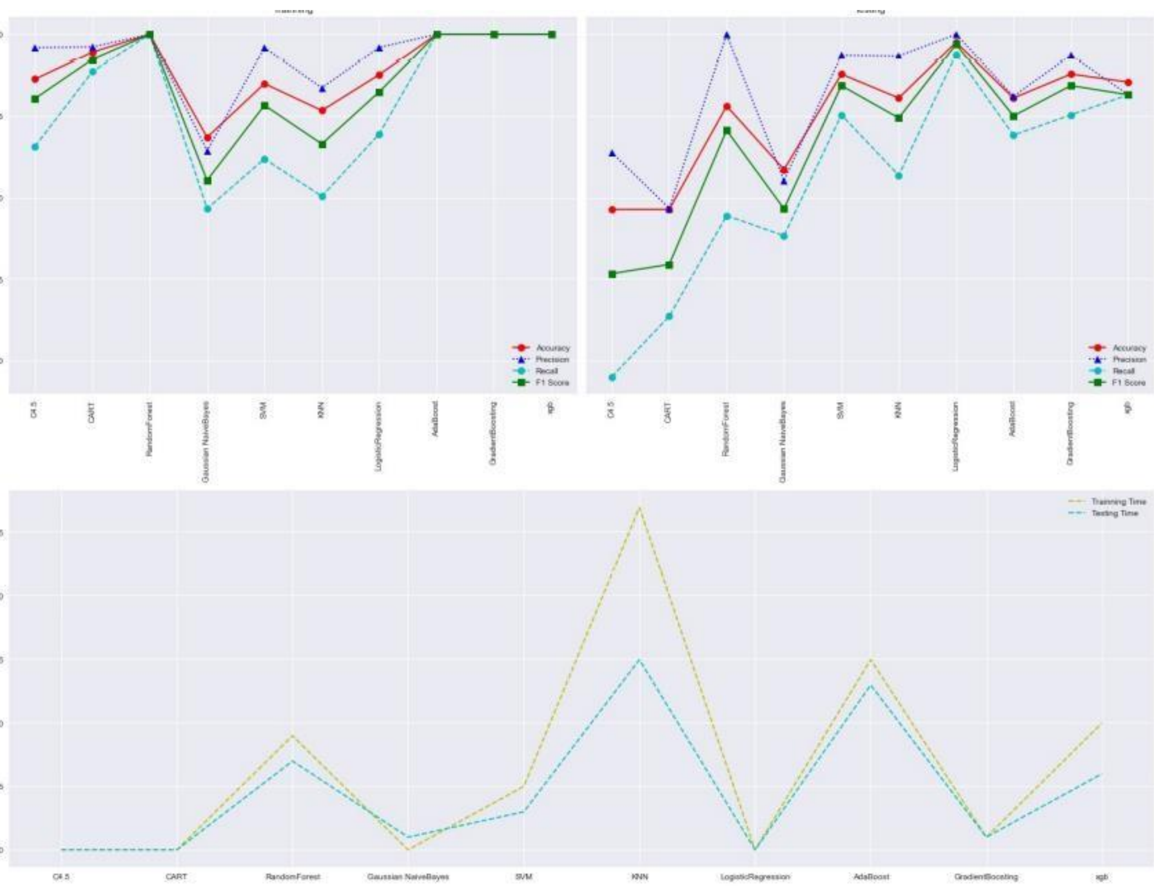
```
In [ ]:
for i in range(10):
    classification_model_report(models[i],model_name[i],0)
```

```
In [ ]:
plt.subplots(ncols=5, nrows=2,figsize=(20,8),sharey=True,sharex=True)
for i in range(1,11):
    plt.subplot(2,5,i)
    sns.heatmap(confusion_matrixs[i-1],annot=True,fmt="d",cbar=False)
    plt.title(model_name[i-1])
```




```
In [ ]:
fig, axs = plt.subplots(ncols=2, nrows=2,figsize=(20,15),sharey=True)
gs = axs[1, 0].get_gridspec()
plt.subplot(2,2,1)
plt.plot(model_name,acc_train,'r',marker='o', markersize=9)
plt.plot(model_name,pres_train,'b:',marker='^', markersize=9)
plt.plot(model_name,rec_train,'c--',marker='o', markersize=9)
plt.plot(model_name,f1_train,'g',marker='s', markersize=9)
plt.xticks(rotation=90)
plt.title("Trainning")
plt.grid(True)
plt.legend(['Accuracy','Precision','Recall','F1 Score'],loc ="lower right")

plt.subplot(2,2,2)
plt.plot(model_name,acc_test,'r',marker='o', markersize=9)
plt.plot(model_name,pres_test,'b:',marker='^', markersize=9)
plt.plot(model_name,rec_test,'c--',marker='o', markersize=9)
plt.plot(model_name,f1_test,'g',marker='s', markersize=9)
plt.xticks(rotation=90)
plt.title("Testing")
plt.grid(True)
plt.legend(['Accuracy','Precision','Recall','F1 Score'],loc ="lower right")
```

```
for ax in axs[1,:]:
    ax.remove()
axbig = fig.add_subplot(gs[1,:])

plt.plot(model_name,train_time,'y--')
plt.plot(model_name,test_time,'c--')
plt.legend(['Trainning Time','Testing Time'],loc ="upper right")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Here, we observe that Logistic Regression performs very well both in terms of time and performance. It has accquired an accuracy of 99.5%

In [ ]:

```
for i in range(len(model_name)):
    print(model_name[i]+" : "+str(acc_test[i]*100))
```

```
C4.5 : 89.26829268292683
CART : 89.26829268292683
RandomForest : 95.60975609756098
Gaussian NaiveBayes : 91.70731707317074
SVM : 97.5609756097561
KNN : 96.09756097560975
LogisticRegression : 99.51219512195122
AdaBoost : 96.09756097560975
GradientBoosting : 97.5609756097561
xgb : 97.07317073170731
```

In [ ]:

```
###Choosing the best models for creating the meta - learning models.
selected_names = []
selected_models = []
selected_acc = []
for i in range(len(model_name)):
    if acc_test[i]>0.95:
        selected_names.append(model_name[i])
        selected_models.append(models[i])
        selected_acc.append(acc_test[i])
for i in range(len(selected_models)):
    print(selected_names[i]+" : "+str(selected_acc[i]*100))
```

```
RandomForest : 95.60975609756098
SVM : 97.5609756097561
KNN : 96.09756097560975
LogisticRegression : 99.51219512195122
AdaBoost : 96.09756097560975
GradientBoosting : 97.5609756097561
xgb : 97.07317073170731
```

Creating Meta - Learning models

In [ ]:

```
def estimate_creator(l):
    estm = []
    for i in l:
        estm.append((selected_names[i],selected_models[i]))
    return estm
```

In [ ]:

```
from sklearn.ensemble import StackingClassifier

l = [[0,1,2,3,4],[3,1,5,6,2],[0,3,5],[0,1,3,5],[0,1,2,3,4,5,6],[3,6,1,2,4],[3,2,1,6,5],[3,1]]

estimator_list = []
for i in range(len(l)):
    estimator_list.append(estimate_creator(l[i]))

# Build stack model
stack_model_list = []
for i in range(len(l)):
    stack_model_list.append(StackingClassifier(estimators=estimator_list[i], final_estimator=LogisticRegression(

stack_model_name = []
for i in range(len(l)):
    stack_model_name.append(str("Meta Model "+str(i+1)))
```

In [ ]:

```
stack_acc_train = []
stack_acc_test = []
stack_pres_train = []
stack_pres_test = []
stack_rec_train = []
stack_rec_test = []
stack_f1_train = []
stack_f1_test = []
stack_train_time = []
stack_test_time = []
stack_confusion_matrixs = []
test_prediction = []
train_prediction = []
```

In [ ]:

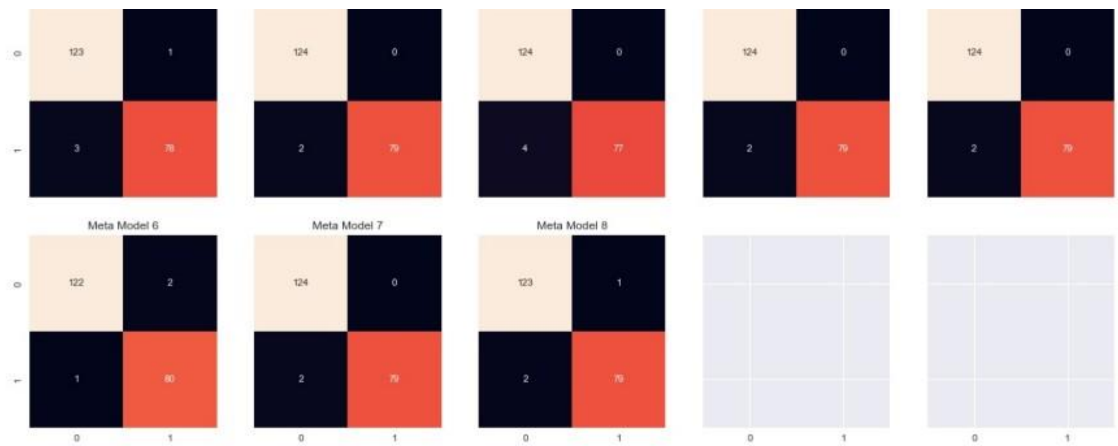
```
def stack_classification_model_report(model,name,n):
  #Fit the model:
  model = model.fit(x_train_12,y_train)

  #Make predictions on training set:
  start_time = time.time()
  pred_train = model.predict(x_train_12)
  end_time = time.time()
  train_time_model = end_time-start_time
  stack_train_time.append(train_time_model)
  train_prediction.append(pred_train)

  start_time = time.time()
  pred_test = model.predict(x_test_12)
  end_time = time.time()
  test_time_model = end_time-start_time
  stack_test_time.append(test_time_model)
  test_prediction.append(pred_test)

  #Print accuracy
  ac_train = accuracy_score(y_train,pred_train)
  ac_test = accuracy_score(y_test,pred_test)
  stack_acc_train.append(ac_train)
  stack_acc_test.append(ac_test)

  #Print precision
  pr_train = precision_score(y_train, pred_train)
  pr_test = precision_score(y_test, pred_test)
  stack_pres_train.append(pr_train)
  stack_pres_test.append(pr_test)

  #Print recall
  re_train = recall_score(y_train, pred_train)
  re_test = recall_score(y_test, pred_test)
  stack_rec_train.append(re_train)
  stack_rec_test.append(re_test)

  #Print f1_score
  f_train = f1_score(y_train, pred_train)
  f_test = f1_score(y_test, pred_test)
  stack_f1_train.append(f_train)
  stack_f1_test.append(f_test)

  #confusion matrix
  cm = confusion_matrix(y_test,pred_test)
  stack_confusion_matrixs.append(cm)

  if n==1:
    print("|| "+name+" ||\n")
    print("---------------------------------------------------------------------------------------------------------
    print("Trainning\n")
    print("Time: ",train_time_model, end=" || ")
    print("Accuracy: ",round(ac_train,5), end=" || ")
    print("Precision: ",round(pr_train,5), end=" || ")
    print("Recall: ",round(re_train,5), end=" || ")
    print("f1_score: ",round(f_train,5))
    print("\n-------------------------------------------------------------------------------------------------------
    print("Testing\n")
    print("Time: ",test_time_model, end=" || ")
    print("Accuracy: ",round(ac_test,5), end=" || ")
    print("Precision: ",round(pr_test,5), end=" || ")
    print("Recall: ",round(re_test,5), end=" || ")
    print("f1_score: ",round(f_test,5))
    print("\n-------------------------------------------------------------------------------------------------------
```

In [ ]:

```
for i in range(len(l)):
    stack_classification_model_report(stack_model_list[i],stack_model_name[i],0)
```

In [ ]:

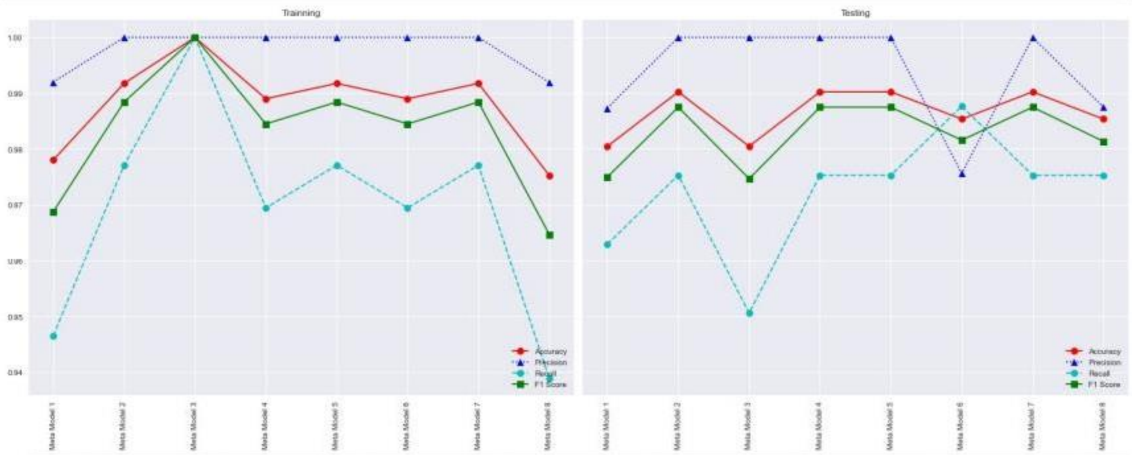
```
plt.subplots(ncols=5, nrows=2,figsize=(20,8),sharey=True,sharex=True)
for i in range(len(l)):
    plt.subplot(2,5,i+1)
    sns.heatmap(stack_confusion_matrixs[i],annot=True,fmt="d",cbar=False)
    plt.title(stack_model_name[i])
```

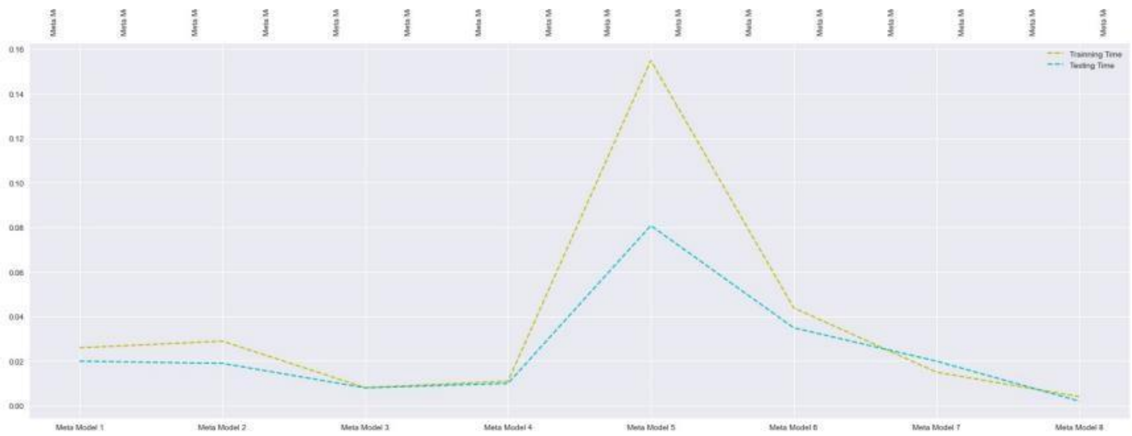


In [ ]:

```
fig, axs = plt.subplots(ncols=2, nrows=2,figsize=(20,15),sharey=True)
gs = axs[1, 0].get_gridspec()
plt.subplot(2,2,1)
plt.plot(stack_model_name,stack_acc_train,'r',marker='o', markersize=9)
plt.plot(stack_model_name,stack_pres_train,'b:',marker='^', markersize=9)
plt.plot(stack_model_name,stack_rec_train,'c--',marker='o', markersize=9)
plt.plot(stack_model_name,stack_f1_train,'g',marker='s', markersize=9)
plt.xticks(rotation=90)
plt.title("Trainning")
plt.grid(True)
plt.legend(['Accuracy','Precision','Recall','F1 Score'],loc ="lower right")

plt.subplot(2,2,2)
```

```
plt.plot(stack_model_name,stack_acc_test,'r',marker='o', markersize=9)
plt.plot(stack_model_name,stack_pres_test,'b:',marker='^', markersize=9)
plt.plot(stack_model_name,stack_rec_test,'c--',marker='o', markersize=9)
plt.plot(stack_model_name,stack_f1_test,'g',marker='s', markersize=9)
plt.xticks(rotation=90)
plt.title("Testing")
plt.grid(True)
plt.legend(['Accuracy','Precision','Recall','F1 Score'],loc ="lower right")

for ax in axs[1,:]:
    ax.remove()
axbig = fig.add_subplot(gs[1,:])

plt.plot(stack_model_name,stack_train_time,'y--')
plt.plot(stack_model_name,stack_test_time,'c--')
plt.legend(['Trainning Time','Testing Time'],loc ="upper right")
plt.grid(True)
plt.tight_layout()
plt.show()
```





[ ]:

```
for i in range(len(stack_model_name)):
    print(stack_model_name[i]+" : "+str(stack_acc_test[i]*100))
```

```
Meta Model 1 : 98.04878048780488
Meta Model 2 : 99.02439024390245
Meta Model 3 : 98.04878048780488
Meta Model 4 : 99.02439024390245
Meta Model 5 : 99.02439024390245
Meta Model 6 : 98.53658536585365
Meta Model 7 : 99.02439024390245
Meta Model 8 : 98.53658536585365
```

Creating ANN Model using Meta Learning Models

[ ]:

```
### Creating dataset for ANN Model
creator = {}
for i in range(len(l)):
    creator[stack_model_name[i]] = train_prediction[i]

df_ann = pd.DataFrame(creator)
df_ann.head()
```

Out[ ]:

	Meta Model 1	Meta Model 2	Meta Model 3	Meta Model 4	Meta Model 5	Meta Model 6	Meta Model 7	Meta Model 8
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0

In [ ]:

```
df_ann.to_csv('train.csv',index=False)
y_train.to_csv('y_train.csv',index=False)
```

In [ ]:

```
### Creating dataset for ANN Model
creator = {}
for i in range(len(l)):
    creator[stack_model_name[i]] = test_prediction[i]

df_ann_test = pd.DataFrame(creator)
df_ann_test.head()
```

Out[ ]:

	Meta Model 1	Meta Model 2	Meta Model 3	Meta Model 4	Meta Model 5	Meta Model 6	Meta Model 7	Meta Model 8
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1

In [ ]:
```
df_ann_test.to_csv('test.csv',index=False)
y_test.to_csv('y_test.csv',index=False)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

In [ ]:
```
# Initialising the ANN
classifier = Sequential()
# Adding the input layer and the first hidden layer
classifier.add(Dense(5, kernel_initializer='uniform', activation='relu', input_dim=len(l)))
# Adding dropout to prevent overfitting
classifier.add(Dropout(0.1))
# Adding the second hidden layer
classifier.add(Dense(3, kernel_initializer='uniform', activation='relu'))
# Adding dropout to prevent overfitting
classifier.add(Dropout(0.1))
# Adding the output layer
classifier.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))

# Compiling the ANN
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

In [ ]:
```
classifier.fit(df_ann.values, y_train, batch_size=100, epochs=150)
```

```
Epoch 1/150
4/4 [==============================] - 1s 7ms/step - loss: 0.6930 - accurac
y: 0.6401
Epoch 2/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6924 - accurac
y: 0.6401
Epoch 3/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6919 - accurac
y: 0.6401
Epoch 4/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6914 - accurac
y: 0.6401
Epoch 5/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6908 - accurac
y: 0.6401
Epoch 6/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6903 - accurac
y: 0.6401
Epoch 7/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6897 - accurac
y: 0.6401
Epoch 8/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6892 - accurac
y: 0.6401
Epoch 9/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6886 - accurac
y: 0.6401
Epoch 10/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6880 - accurac
y: 0.6401
Epoch 11/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6875 - accurac
y: 0.6401
Epoch 12/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6870 - accurac
y: 0.6401
Epoch 13/150
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.6864 - accurac
y: 0.6401
Epoch 14/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6858 - accurac
y: 0.6401
Epoch 15/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6851 - accurac
y: 0.6401
Epoch 16/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6846 - accurac
y: 0.6401
Epoch 17/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6839 - accurac
y: 0.6401
Epoch 18/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6834 - accurac
y: 0.6401
Epoch 19/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6827 - accurac
y: 0.6401
Epoch 20/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6819 - accurac
y: 0.6401
Epoch 21/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6812 - accurac
y: 0.6401
Epoch 22/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6804 - accurac
y: 0.6401
Epoch 23/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6794 - accurac
y: 0.6401
Epoch 24/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6782 - accurac
y: 0.6401
Epoch 25/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6768 - accurac
y: 0.6401
Epoch 26/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6757 - accurac
y: 0.6401
Epoch 27/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6743 - accurac
y: 0.6401
Epoch 28/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6731 - accurac
y: 0.6401
Epoch 29/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6717 - accurac
y: 0.6401
Epoch 30/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6699 - accurac
y: 0.6401
Epoch 31/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6689 - accurac
y: 0.6401
Epoch 32/150
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.6680 - accurac
y: 0.6401
Epoch 33/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6660 - accurac
y: 0.6401
Epoch 34/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6655 - accurac
y: 0.6401
Epoch 35/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6628 - accurac
y: 0.6401
Epoch 36/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6609 - accurac
y: 0.6401
Epoch 37/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6595 - accurac
y: 0.6401
Epoch 38/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6583 - accurac
y: 0.6401
Epoch 39/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6566 - accurac
y: 0.6401
Epoch 40/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6548 - accurac
y: 0.6401
Epoch 41/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6540 - accurac
y: 0.6401
Epoch 42/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6511 - accurac
y: 0.6401
Epoch 43/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6504 - accurac
y: 0.6401
Epoch 44/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6481 - accurac
y: 0.6401
Epoch 45/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6466 - accurac
y: 0.6401
Epoch 46/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6448 - accurac
y: 0.6401
Epoch 47/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6429 - accurac
y: 0.6401
Epoch 48/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6400 - accurac
y: 0.6401
Epoch 49/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6381 - accurac
y: 0.6401
Epoch 50/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6360 - accurac
y: 0.6401
Epoch 51/150
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.6336 - accurac
y: 0.6401
Epoch 52/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6307 - accurac
y: 0.6401
Epoch 53/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6281 - accurac
y: 0.6401
Epoch 54/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6220 - accurac
y: 0.6401
Epoch 55/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6218 - accurac
y: 0.6401
Epoch 56/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6198 - accurac
y: 0.6401
Epoch 57/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6151 - accurac
y: 0.6401
Epoch 58/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6122 - accurac
y: 0.6401
Epoch 59/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6066 - accurac
y: 0.6401
Epoch 60/150
4/4 [==============================] - 0s 2ms/step - loss: 0.6007 - accurac
y: 0.6401
Epoch 61/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5961 - accurac
y: 0.6401
Epoch 62/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5901 - accurac
y: 0.6401
Epoch 63/150
4/4 [==============================] - 0s 3ms/step - loss: 0.5824 - accurac
y: 0.6401
Epoch 64/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5793 - accurac
y: 0.6401
Epoch 65/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5726 - accurac
y: 0.6896
Epoch 66/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5645 - accurac
y: 0.8736
Epoch 67/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5560 - accurac
y: 0.8819
Epoch 68/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5478 - accurac
y: 0.9038
Epoch 69/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5405 - accurac
y: 0.9038
Epoch 70/150
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.5286 - accurac
y: 0.9258
Epoch 71/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5200 - accurac
y: 0.9478
Epoch 72/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5161 - accurac
y: 0.9505
Epoch 73/150
4/4 [==============================] - 0s 2ms/step - loss: 0.5038 - accurac
y: 0.9478
Epoch 74/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4963 - accurac
y: 0.9505
Epoch 75/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4831 - accurac
y: 0.9615
Epoch 76/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4743 - accurac
y: 0.9643
Epoch 77/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4593 - accurac
y: 0.9560
Epoch 78/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4511 - accurac
y: 0.9588
Epoch 79/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4473 - accurac
y: 0.9368
Epoch 80/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4276 - accurac
y: 0.9643
Epoch 81/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4306 - accurac
y: 0.9423
Epoch 82/150
4/4 [==============================] - 0s 2ms/step - loss: 0.4087 - accurac
y: 0.9505
Epoch 83/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3937 - accurac
y: 0.9698
Epoch 84/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3948 - accurac
y: 0.9533
Epoch 85/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3727 - accurac
y: 0.9643
Epoch 86/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3767 - accurac
y: 0.9451
Epoch 87/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3736 - accurac
y: 0.9478
Epoch 88/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3632 - accurac
y: 0.9423
Epoch 89/150
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.3672 - accurac
y: 0.9423
Epoch 90/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3356 - accurac
y: 0.9615
Epoch 91/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3493 - accurac
y: 0.9368
Epoch 92/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3113 - accurac
y: 0.9670
Epoch 93/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3213 - accurac
y: 0.9560
Epoch 94/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3082 - accurac
y: 0.9505
Epoch 95/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3111 - accurac
y: 0.9451
Epoch 96/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2994 - accurac
y: 0.9533
Epoch 97/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3062 - accurac
y: 0.9396
Epoch 98/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3064 - accurac
y: 0.9423
Epoch 99/150
4/4 [==============================] - 0s 2ms/step - loss: 0.3115 - accurac
y: 0.9258
Epoch 100/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2838 - accurac
y: 0.9560
Epoch 101/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2797 - accurac
y: 0.9505
Epoch 102/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2689 - accurac
y: 0.9533
Epoch 103/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2829 - accurac
y: 0.9423
Epoch 104/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2589 - accurac
y: 0.9643
Epoch 105/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2552 - accurac
y: 0.9560
Epoch 106/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2615 - accurac
y: 0.9478
Epoch 107/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2643 - accurac
y: 0.9368
Epoch 108/150
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.2689 - accurac
y: 0.9451
Epoch 109/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2331 - accurac
y: 0.9615
Epoch 110/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2453 - accurac
y: 0.9505
Epoch 111/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2255 - accurac
y: 0.9643
Epoch 112/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2330 - accurac
y: 0.9560
Epoch 113/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2395 - accurac
y: 0.9423
Epoch 114/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2003 - accurac
y: 0.9835
Epoch 115/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2274 - accurac
y: 0.9478
Epoch 116/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2078 - accurac
y: 0.9670
Epoch 117/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2056 - accurac
y: 0.9643
Epoch 118/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2108 - accurac
y: 0.9560
Epoch 119/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2244 - accurac
y: 0.9451
Epoch 120/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2094 - accurac
y: 0.9588
Epoch 121/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2083 - accurac
y: 0.9533
Epoch 122/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2075 - accurac
y: 0.9451
Epoch 123/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2056 - accurac
y: 0.9478
Epoch 124/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2016 - accurac
y: 0.9505
Epoch 125/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1934 - accurac
y: 0.9643
Epoch 126/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1828 - accurac
y: 0.9643
Epoch 127/150
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.1862 - accurac
y: 0.9588
Epoch 128/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1902 - accurac
y: 0.9615
Epoch 129/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1829 - accurac
y: 0.9615
Epoch 130/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2115 - accurac
y: 0.9286
Epoch 131/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1795 - accurac
y: 0.9588
Epoch 132/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2081 - accurac
y: 0.9341
Epoch 133/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1569 - accurac
y: 0.9670
Epoch 134/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2016 - accurac
y: 0.9313
Epoch 135/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1853 - accurac
y: 0.9478
Epoch 136/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1685 - accurac
y: 0.9588
Epoch 137/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1767 - accurac
y: 0.9505
Epoch 138/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1546 - accurac
y: 0.9643
Epoch 139/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1864 - accurac
y: 0.9451
Epoch 140/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1606 - accurac
y: 0.9588
Epoch 141/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1652 - accurac
y: 0.9505
Epoch 142/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1720 - accurac
y: 0.9505
Epoch 143/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1685 - accurac
y: 0.9505
Epoch 144/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1668 - accurac
y: 0.9478
Epoch 145/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1751 - accurac
y: 0.9368
Epoch 146/150
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.1635 - accurac
y: 0.9478
Epoch 147/150
4/4 [==============================] - 0s 4ms/step - loss: 0.1634 - accurac
y: 0.9396
Epoch 148/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1546 - accurac
y: 0.9588
Epoch 149/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1391 - accurac
y: 0.9698
Epoch 150/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1595 - accurac
y: 0.9478
```

Out[ ]: `<keras.callbacks.History at 0x275917dd970>`

In [ ]:

```
pred_train = classifier.predict(df_ann.values)
pred_train = (pred_train > 0.5)
```

In [ ]:

```
pred_test = classifier.predict(df_ann_test.values)
pred_test = (pred_test > 0.5)
```

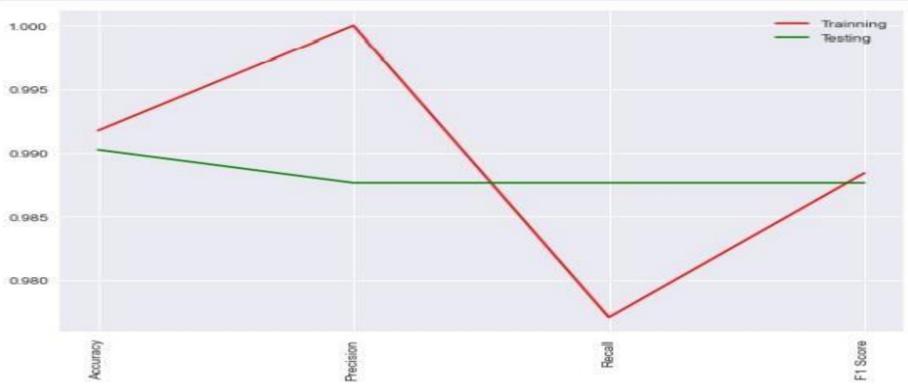
In [ ]:

```
ann_acc_train = accuracy_score(y_train,pred_train)
ann_acc_test = accuracy_score(y_test,pred_test)
ann_pr_train = precision_score(y_train, pred_train)
ann_pr_test = precision_score(y_test, pred_test)
ann_re_train = recall_score(y_train, pred_train)
ann_re_test = recall_score(y_test, pred_test)
ann_f_train = f1_score(y_train, pred_train)
ann_f_test = f1_score(y_test, pred_test)

print("Accuracy Training: ",ann_acc_train)
print("Accuracy Testing: ",ann_acc_test)
```

```
Accuracy Training:  0.9917582417582418
Accuracy Testing:  0.9902439024390244
```

In [ ]:

```
metrics_name = ['Accuracy','Precision','Recall','F1 Score']
train_metrics = [ann_acc_train, ann_pr_train, ann_re_train, ann_f_train]
test_metrics = [ann_acc_test, ann_pr_test, ann_re_test, ann_f_test]
plt.plot(metrics_name,train_metrics,'r')
plt.plot(metrics_name,test_metrics,'g')
plt.xticks(rotation=90)
plt.legend(['Trainning','Testing'],loc ="upper right")
plt.tight_layout()
plt.show()
```



In [ ]:

```
creator_train = {}
creator_train['Model'] = model_name
creator_train['Accuracy'] = acc_train
creator_train['Precision'] = pres_train
creator_train['Recall'] = rec_train
creator_train['F1 Score'] = f1_train
training_ML_models = pd.DataFrame(creator_train)
training_ML_models
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1 Score
0	C4.5	0.972527	0.991870	0.931298	0.960630
1	CART	0.989011	0.992248	0.977099	0.984615
2	RandomForest	1.000000	1.000000	1.000000	1.000000
3	Gaussian NaiveBayes	0.936813	0.928571	0.893130	0.910506
4	SVM	0.969780	0.991803	0.923664	0.956522
5	KNN	0.953297	0.967213	0.900763	0.932806
6	LogisticRegression	0.975275	0.991935	0.938931	0.964706
7	AdaBoost	1.000000	1.000000	1.000000	1.000000
8	GradientBoosting	1.000000	1.000000	1.000000	1.000000
9	xgb	1.000000	1.000000	1.000000	1.000000

In [ ]:

```
training_ML_models.to_csv('compare_train.csv',index=False)
```

In [ ]:

```
creator_test = {}
creator_test['Model'] = model_name
creator_test['Accuracy'] = acc_test
creator_test['Precision'] = pres_test
creator_test['Recall'] = rec_test
creator_test['F1 Score'] = f1_test
testing_ML_models = pd.DataFrame(creator_test)
testing_ML_models
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1 Score
0	C4.5	0.892683	0.927536	0.790123	0.853333
1	CART	0.892683	0.893333	0.827160	0.858974
2	RandomForest	0.956098	1.000000	0.888889	0.941176
3	Gaussian NaiveBayes	0.917073	0.910256	0.876543	0.893082
4	SVM	0.975610	0.987179	0.950617	0.968553
5	KNN	0.960976	0.986667	0.913580	0.948718
6	LogisticRegression	0.995122	1.000000	0.987654	0.993789
7	AdaBoost	0.960976	0.962025	0.938272	0.950000
8	GradientBoosting	0.975610	0.987179	0.950617	0.968553
9	xgb	0.970732	0.962963	0.962963	0.962963

In [ ]:

```
testing_ML_models.to_csv('compare_test.csv',index=False)
```

In [ ]:

```
stack_creator_train = {}
stack_creator_train['Model'] = stack_model_name
stack_creator_train['Accuracy'] = stack_acc_train
stack_creator_train['Precision'] = stack_pres_train
stack_creator_train['Recall'] = stack_rec_train
stack_creator_train['F1 Score'] = stack_f1_train
stack_training_ML_models = pd.DataFrame(stack_creator_train)
stack_training_ML_models
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1 Score
0	Meta Model 1	0.978022	0.992000	0.946565	0.968750
1	Meta Model 2	0.991758	1.000000	0.977099	0.988417
2	Meta Model 3	1.000000	1.000000	1.000000	1.000000
3	Meta Model 4	0.989011	1.000000	0.969466	0.984496
4	Meta Model 5	0.991758	1.000000	0.977099	0.988417
5	Meta Model 6	0.989011	1.000000	0.969466	0.984496
6	Meta Model 7	0.991758	1.000000	0.977099	0.988417
7	Meta Model 8	0.975275	0.991935	0.938931	0.964706

In [ ]:

```
stack_training_ML_models.to_csv('compare_train_stack.csv',index=False)
```

In [ ]:

```
stack_creator_test = {}
stack_creator_test['Model'] = stack_model_name
stack_creator_test['Accuracy'] = stack_acc_test
stack_creator_test['Precision'] = stack_pres_test
stack_creator_test['Recall'] = stack_rec_test
stack_creator_test['F1 Score'] = stack_f1_test
stack_testing_ML_models = pd.DataFrame(stack_creator_test)
stack_testing_ML_models
```

Out[ ]:

	Model	Accuracy	Precision	Recall	F1 Score
0	Meta Model 1	0.980488	0.987342	0.962963	0.975000
1	Meta Model 2	0.990244	1.000000	0.975309	0.987500
2	Meta Model 3	0.980488	1.000000	0.950617	0.974684
3	Meta Model 4	0.990244	1.000000	0.975309	0.987500
4	Meta Model 5	0.990244	1.000000	0.975309	0.987500
5	Meta Model 6	0.985366	0.975610	0.987654	0.981595
6	Meta Model 7	0.990244	1.000000	0.975309	0.987500
7	Meta Model 8	0.985366	0.987500	0.975309	0.981366

In [ ]:

```
stack_testing_ML_models.to_csv('compare_test_stack.csv',index=False)
```

In [ ]:

```
ann_creator = {}
ann_creator['ANN'] = ['Trainning','Testing']
ann_creator['Accuracy'] = [ann_acc_train,ann_acc_test]
ann_creator['Precision'] = [ann_pr_train,ann_pr_test]
ann_creator['Recall'] = [ann_re_train,ann_re_test]
ann_creator['F1 Score'] = [ann_f_train,ann_f_test]
ann_models = pd.DataFrame(ann_creator)
ann_models
```

Out[ ]:

	ANN	Accuracy	Precision	Recall	F1 Score
0	Trainning	0.991758	1.000000	0.977099	0.988417
1	Testing	0.990244	0.987654	0.987654	0.987654

COMPARATIVE STUDY / RESULTS AND DISCUSSION

Parameters	**[15]**	**[17]**	**Proposed System**
Models Used	SVM	ANN (back propagation)	Random Forest, Logistic Regression,SVM,KNN,Gradient Boosting,xgb,ANN
Training/Testing Ratio	70:30	79:21	64:36
Training Accuracy	-	-	0.991758
Testing Accuracy	97.9%	99.4624%	0.990244
Training Precision	-	-	1.000000
Testing Precision	97.9%	-	0.987654
Training Recall	-	-	0.977099
Testing Recall	97.9%	-	0.987654
Training F1-Score	-	-	0.988417
Testing F1-Score	-	-	0.987654

Future works

we will hyper tune various machine learning models and combine them in various permutation to create new meta models with higher accuracy which will in turn increase the accuracy of the hybrid model.

REFERENCES:

[1] .M S. Yarabarla, L. K. Ravi, and A. Sivasangari, ‘‘Breast cancer prediction via machine learning,’’ in Proc. 3rd Int. Conf. Trends Electron. Informat. (ICOEI), Apr. 2019, pp. 121–124.

[2].S Alghunaim and H. H. Al-Baity, ‘‘On the scalability of machine learning algorithms for breast cancer prediction in big data context,’’ IEEE Access, vol. 7, pp. 91535–91546, 2019.

[3]. A. A. Bataineh, ‘‘A comparative analysis of nonlinear machine learning algorithms for breast cancer detection,’’ Int. J. Mach. Learn. Comput., vol. 9, no. 3, pp. 248–254, Jun. 2019.

[4]. E A. Bayrak, P. Kirci, and T. Ensari, ‘‘Comparison of machine learning methods for breast cancer diagnosis,’’ in Proc. Sci. Meeting Elect.- Electron. Biomed. Eng. Comput. Sci. (EBBT), Apr. 2019, pp. 1–3.

[5]. H. Dhahri, E. Al Maghayreh, A. Mahmood, W. Elkilani, and M. Faisal Nagi, ‘‘Automated breast cancer diagnosis based on machine learning algorithms,’’ J. Healthcare Eng., vol. 2019, pp. 1–11, Nov. 2019.

[6]. M. K. Keles, ‘‘Breast cancer prediction and detection using data mining classification algorithms: A comparative study,’’ Tehnički Vjesnik, vol. 26, no. 1, pp. 149–155, 2019.

[7]. K. Williams, P. A. Idowu, J. A. Balogun, and A. I. Oluwaranti, ‘‘Breast cancer risk prediction using data mining classification techniques,’’ Trans. Netw. Commun., vol. 3, no. 2, p. 1, Apr. 2015.

[8]. B Padmapriya and T. Velmurugan, ‘‘Classification algorithm based analysis of breast cancer data,’’ Int. J. Data Mining Techn. Appl., vol. 5, no. 1, pp. 43–49, Jun. 2016.

[9]. H. Asri, H. Mousannif, H. Al Moatassime, and T. Noel, ‘‘Using machine learning algorithms for breast cancer risk prediction and diagnosis,’’ Procedia Comput. Sci., vol. 83, pp. 1064–1069, Jan. 2016.

[10].U. Ojha and S. Goel, ‘‘A study on prediction of breast cancer recurrence using data mining techniques,’’ in Proc. 7th Int. Conf. Cloud Comput., Data Sci. Eng.-Confluence, Jan. 2017, pp. 527–530.

[11]. S. K. Maliha, R. R. Ema, S. K. Ghosh, H. Ahmed, M. R. J. Mollick, and T. Islam, ‘‘Cancer disease prediction using naive Bayes,K-nearest neighbor and J48 algorithm,’’ in Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT), Jul. 2019, pp. 1–7.

[12].M. H. Memon, J. P. Li, A. U. Haq, M. H. Memon, and W. Zhou, ‘‘Breast cancer detection in the IOT health environment using modified recursive feature selection,’’ Wireless Commun. Mobile Comput., vol. 2019, pp. 1–19, Nov. 2019.

[13]. D. A. Omondiagbe, S. Veeramani, and A. S. Sidhu, ‘‘Machine learning classification techniques for breast cancer diagnosis,’’ IOP Conf. Ser., Mater. Sci. Eng., vol. 495, Jun. 2019, Art. no. 012033.

[14]. A Bharat, N. Pooja, and R. A. Reddy, ‘‘Using machine learning algorithms for breast cancer risk prediction and diagnosis,’’ in Proc. 3rd Int. Conf. Circuits, Control, Commun. Comput. (IC), Oct. 2018, pp. 1–4.

[15]. Y. Khourdifi and M. Bahaj, ‘‘Applying best machine learning algorithms for breast cancer prediction and classification,’’ in Proc. Int. Conf. Electron., Control, Optim. Comput. Sci. (ICECOCS), Dec. 2018, pp. 1–5.

[16].AA Said, L. A. Abd-Elmegid, S. Kholeif, and A. Abdelsamie, ‘‘Classification based on clustering model for predicting main outcomes of breast cancer using hyper-parameters optimization,’’ Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 12, pp. 268–273, 2018

[17]. P. Singhal and S. Pareek, ‘‘Artificial neural network for prediction of breast cancer,’’ in Proc. 2nd Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud)(I-SMAC), 2018, pp. 464–468.

[18]. A. A. Ibrahim, A. I. Hashad, and N. E. M. Shawky, ‘‘A comparison of open source data mining tools for breast cancer classification,’’ in Handbook of Research on Machine Learning Innovations and Trends. Hershey, PA, USA: IGI Global, 2017, pp. 636–651.

[19].Macías-García, L., Luna-Romera, J. M., García-Gutiérrez, J., Martínez-Ballesteros, M., Riquelme-Santos, J. C., & González-Cámpora, R. (2017). A study of the suitability of autoencoders for preprocessing data in breast cancer experimentation. Journal of Biomedical Informatics, 72, 33–44. doi:10.1016/j.jbi.2017.06.020

[20]. Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. Computational and Structural Biotechnology Journal, 13, 8–17. doi:10.1016/j.csbj.2014.11.005

[21]. Saba, T. (2020). Recent advancement in cancer detection using machine learning: Systematic survey of decades, comparisons and challenges. Journal of Infection and Public Health.

[22]. Nonita Sharma, Monika Mangla, Sachi Nandan Mohanty, Suneeta Satpaty, "A Stochastic Neighbor Embedding Approach for Cancer Prediction", Emerging Smart Computing and Informatics (ESCI) 2021 International Conference on, pp. 599-603, 2021

[23]. Doddipalli, Lavanya & Rani, K.. (2011). Analysis of feature selection with classification: Breast cancer datasets. Indian Journal of Computer Science and Engineering (IJCSE). 2. 756-763.

[24]. Mehmet Fatih Akay, Support vector machines combined with feature selection for breast cancer diagnosis, Expert Systems with Applications, Volume 36, Issue 2, Part 2, 2009, Pages 3240-3247, ISSN 0957-4174.

[25]. Karabatak, M. (2015). A new classifier for breast cancer detection based on Naïve Bayesian. Measurement, 72, 32–36.

[26]. Alyami, R., Alhajjaj, J., Alnajrani, B., Elaalami, I., Alqahtani, A., Aldhafferi, N., … Olatunji, S. O. (2017). Investigating the effect of correlation based feature selection on breast cancer diagnosis using artificial neural network and support vector machines. 2017

[27]. Islam, M. M., Haque, M. R., Iqbal, H., Hasan, M. M., Hasan, M., & Kabir, M. N. (2020). Breast Cancer Prediction: A Comparative Study Using Machine Learning Techniques

[28]. Li J, Zhou Z, Dong J, Fu Y, Li Y, Luan Z, et al. (2021) Predicting breast cancer 5-year survival using machine learning: A systematic review.

[29]. A. Kumar, R. Patra and A. Ghosh, "Model Selection for Predicting Breast Cancer using Supervised Machine Learning Algorithms," 2020 IEEE 1st International Conference for Convergence in Engineering (ICCE), 2020, pp. 320-324, doi: 10.1109/ICCE50343.2020.9290578.

[30].Naveen, R. K. Sharma and A. Ramachandran Nair, "Efficient Breast Cancer Prediction Using Ensemble Machine Learning Models," 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2019, pp. 100-104, doi: 10.1109/RTEICT46194.2019.90169