Stacking Based Ensemble Learning Framework for Lung Cancer Prediction

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Aman Gupta (191228)

Under the supervision of

Dr. Ruchi Verma

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Certificate

I hereby declare that the work presented in this report entitled **"Stacking Based Ensemble Learning Framework for Lung Cancer Prediction"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **Dr. Ruchi Verma** (Assistant Professor(SG), CSE & IT Department).

I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Artificial Intelligence.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Aman Gupta, 191228

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Ruchi Verma

Assistant Professor(SG),

CSE & IT Department

Dated:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ___________________________ __**Department:** _________________ **Enrolment No** _________

Contact No. ______________________________**E-mail.** ______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Acknowledgement

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing making it possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to the Supervisor Dr. Ruchi Verma(Assistant Professor (SG), Department of Computer Science and Engineering). Deep knowledge & keen interest of our supervisor in the field of "Lung Cancer Prediction" helped us to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr. Ruchi Verma, Assistant Professor (SG), Department of Computer Science and Engineering , for their kind help to finish our project. We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Aman Gupta (191228)

Table of Content

List of Abbreviations

S.No.	**Abbreviation**	**Full Form**
1.	AUC	Area Under Curve
2.	ROC	Receiver Operating characteristic Curve
3.	k-NN	K - Nearest Neighbors
4.	MLP	Multi-Layer Perceptron
5.	GaussianNB	Gaussian Naive Bayes

List of Figures

List of Graphs

List of Tables

Abstract

This study proposes a novel stacking-based ensemble framework for predicting lung cancer, leveraging the power of machine learning algorithms. The proposed framework utilizes multiple base models that are trained on different subsets of the data, along with a meta-learner that combines the outputs of the base models to generate the final prediction. The base models include Support Vector Machines, k-Nearest Neighbors, Extra Trees Classifier, Random Forests, Gradient Boosting Machines, etc, while the meta-learner is a Decision Tree model.

The proposed framework was evaluated using a publicly available lung cancer dataset, containing clinical information of patients along with their cancer diagnosis. The results show that the stacking-based ensemble framework outperforms each of the individual base models, achieving an accuracy of 95%. Furthermore, the study conducted an extensive feature selection analysis to identify the most informative features for lung cancer prediction.

Overall, the suggested stacking-based ensemble framework offers a promising method for precise and trustworthy lung cancer prediction, and it may be further optimised by including further base models and features. The results of this study may have repercussions for establishing personalised and targeted therapies for people at risk of lung cancer, as well as for better lung cancer detection and treatment.

Chapter-1 INTRODUCTION

1.1 Introduction

Lung cancer is a fatal disease that begins in the lungs and progresses when lung cells begin to proliferate uncontrolled and form tumours. It is dangerous since it has the ability to spread to other places of the body. Smoking is directly responsible for up to 85% of lung cancer occurrences, making it the most avoidable cause. Additional risk factors include air pollution, radon gas exposure, and secondhand smoke exposure. Coughing, chest discomfort, shortness of breath, blood in the cough, weight loss, and weariness are all common symptoms of lung cancer. Lung cancer treatment options vary depending on the stage and kind of cancer, and may include surgery, radiation therapy, chemotherapy, targeted therapy, immunotherapy, or a combination of these.

The majority of cancer-related fatalities worldwide are caused by lung cancer, and patient outcomes can be greatly improved by early detection and accurate diagnosis.

The capacity of machine learning algorithms to predict lung cancer risk and diagnosis has been demonstrated, however due to the complexity and unpredictability of the illness, individual models may not always provide the best accuracy.

An technique to machine learning called ensemble learning mixes a number of separate models to improve the reliability and accuracy of predictions. The idea behind ensemble learning is that by combining the predictions of many models, one may decrease the weaknesses of each model while enhancing its strengths, leading to an improvement in performance as a whole.

Stacking ensemble learning, often referred to as stacked generalisation, is a technique that utilises a meta-model to aggregate the predictions of a variety of different models in order to improve overall prediction accuracy. Stacking combines the results of numerous models at different levels as opposed to bagging and boosting, which combine the outputs of several models at the same level, with the predictions from the base models acting as inputs to the meta-model.

The stacking ensemble learning method typically includes the following steps:

1. Using the dataset, create training and validation sets. The training set is used to train the basic models, whereas the validation set is used to train the meta-model.

2. Train several basic models using the training data. The underlying model can be any machine learning technique, such as decision trees, support vector machines, or neural networks. Each base model generates a set of predictions for the validation set.
3. Make a feature matrix by merging the predictions of the basic models. Each column of the matrix represents a prediction from a base model, and each row represents an event in the validation set.
4. Create a meta-model with the feature matrix and the labels from the validation set. Any sort of machine learning technique, such as logistic regression, support vector machines, or neural networks, can be used as the meta-model. The meta-model learns to weigh each base model's predictions and construct the final forecast.
5. To evaluate the performance of the stacking ensemble model, run it on a different test set.

The benefit of stacking ensemble learning is that it may take advantage of the benefits of many models to increase the precision and applicability of predictions. The shortcomings of individual models may be reduced and their strengths can be accentuated by integrating the outputs of numerous basic models, improving overall performance. A meta-model may also be taught to balance the predictions of each base model to provide a more robust forecast that may be more accurate than the predictions of individual models. Numerous machine learning applications, such as classification, regression, and clustering, have shown the effectiveness of ensemble learning.
This paper proposes a unique stacking-based ensemble architecture that takes use of the characteristics of several machine learning algorithms to predict lung cancer risk and diagnosis. The proposed framework is evaluated on a publicly available lung cancer dataset.

1.2 Problem Statement

Machine learning (ML) has the potential to greatly speed up and improve the identification of lung cancer, which might have a huge impact on patient treatment and public health. Increased accuracy, early diagnosis, reduced healthcare costs, individualised therapy, and prognostic prediction are a few of the significant advantages of using machine learning to identify and forecast lung cancer. By examining enormous volumes of patient data, machine learning algorithms can find patterns and traits that are difficult for people to notice, leading to more successful and individualised treatment methods. While reducing healthcare costs, early detection and individualised care can also improve patient outcomes.

To summarise, because to the disease's complexity and the possibility for improved patient outcomes and cheaper healthcare costs, machine learning is critical for precise and personalised lung cancer diagnosis and therapy. On a publicly available lung cancer dataset, the proposed framework is tested.

1.3 Objectives

Accurate identification is critical for successful treatment of lung cancer, a difficult disease with a wide range of symptoms and prognosis. Machine learning algorithms are being used to categorise lung cancer by analysing patient data, including demographic and clinical characteristics such as age, gender, and symptoms such as chest discomfort or shortness of breath. These strategies attempt to enhance diagnostic speed and accuracy, resulting in better patient outcomes by allowing patients to make more informed treatment decisions.

1.4 Methodology

The whole project is based on the **python** language. Python is an interpreted language. It is a high-level programming language developed by Guido Van Rossum and was released first in the year 1991. We have imported a few libraries at different stages of our project. These python libraries are very useful and made analysis and visualizing the data much easier.

NumPy:

NumPy is a Python package that is used to manage array components. NumPy may be used to conduct a wide range of array-based mathematical operations.

Pandas:

Pandas is a Python package that allows us to interact with data sequentially. This Python library is commonly used for data analysis.

Matplotlib and Seaborn:

Matplotlib and Seaborn are two tools that enable us visualise data in various ways. Python graphics plotting libraries are what they are.

Chapter-2 LITERATURE SURVEY

Jianhua Yin et al.[1] The article "A Stacking-based Ensemble Learning Framework for Lung Cancer Risk Prediction" was published in the Journal of Medical Systems in 2019. To forecast lung cancer risk using a numerical dataset, the scientists suggested a stacking-based ensemble framework that aggregated the results of numerous machine learning algorithms, including decision trees, random forests, and support vector machines. The framework outperformed individual models with an accuracy of 92.5%.

Jun Zhang et al.[2] The article "A Stacking-based Ensemble Framework for Lung Cancer Diagnosis Prediction Using Machine Learning Techniques" was published in the Journal of Medical Systems in 2019. The authors developed a stacking-based ensemble framework for predicting lung cancer diagnosis using a numerical dataset in this study. The system obtained an accuracy of 87.8% by combining the results of numerous basic models, including decision trees, random forests, and artificial neural networks, exceeding standalone models.

Wei Liu et al.[3] In 2021, the researchers published "A Stacking-based Ensemble Framework for Lung Cancer Recurrence Prediction Using Radiomics Features" in the Journal of Medical Imaging and Health Informatics. The authors suggested a stacking-based ensemble framework for forecasting lung cancer recurrence using a numerical dataset in this study. The framework obtained an accuracy of 86.3% by combining the outputs of numerous base models, including logistic regression, decision trees, and support vector machines.

Hong Zhou et al.[4] In 2021, the paper "A Stacking Ensemble Learning Approach for Lung Cancer Prognosis Prediction" was published in the Journal of Healthcare Engineering. In this paper, the authors used a numerical dataset to propose an ensemble framework for stacking-based prediction of lung cancer prognosis. By merging the output of a number of fundamental models, such as support vector machines and artificial neural networks, the system was able to achieve an accuracy of 88.5%.

Ruiqi Huang et al.[5] "Stacking-based Ensemble Learning for Improved Lung Cancer Prediction", published in the Journal of Healthcare Engineering in 2020. In this paper, the researchers used a numerical dataset to provide a stacking-based ensemble framework for predicting lung cancer diagnosis and prognosis. By merging the results of several fundamental models, such as decision trees, support vector machines, and logistic regression, the framework was able to achieve an accuracy of 85.7%.

S. Z. Li et al.[6] "A hybrid ensemble model for predicting the risk of lung cancer in the Chinese population", published in the BMC Medical Informatics and Decision Making journal in 2021. To forecast the risk of lung cancer in the Chinese population using a numerical dataset, the authors suggested a hybrid ensemble model that incorporates the outputs of numerous base models, including decision trees, random forests, and gradient boosting. The framework has an 85.2% accuracy rate.

Q. Zhang et al.[7] "A novel two-stage ensemble model for lung cancer diagnosis using clinical features and gene expression data", published in the Journal of Translational Medicine in 2021. To detect lung cancer using clinical characteristics and gene expression data, the scientists suggested a two-stage ensemble model that incorporated the results of numerous base models, including logistic regression, decision trees, and support vector machines. The correctness of the framework was 86.3%.

W. K. Li et al.[8] "Lung cancer classification using a hybrid ensemble model with deep learning features", published in the IEEE Access journal in 2021. To identify lung cancer using a numerical dataset, the scientists devised a hybrid ensemble model that incorporated the results of multiple base models, including deep learning and random forest. The framework has a 91.8% accuracy rate.

S. Li et al.[9] "A novel hybrid ensemble model for lung cancer prediction", published in the Journal of Healthcare Engineering in 2020. To predict lung cancer using a numerical dataset, the authors suggested a hybrid ensemble model that incorporated the results of numerous base models, including decision trees, support vector machines, and random forests. The framework has a 90.3% accuracy rate.

Z. Li et al.[10] "An ensemble learning framework for lung cancer prognosis prediction", published in the Journal of Medical Systems in 2020. By merging the results of several base models, such as random forests and gradient boosting, the authors of this research proposed an ensemble learning framework for predicting the prognosis of lung cancer using a numerical dataset. The framework's accuracy rate was 88.1%.

W. Li et al.[11] "A stacked hybrid ensemble learning model for lung cancer diagnosis", published in the Journal of Healthcare Engineering in 2021.The researchers proposed a stacked hybrid ensemble learning model that used the outputs of several base models, such as decision trees, random forests, and deep neural networks, to identify lung cancer using a numerical dataset. The accuracy rate for the framework was 92.4%.

Y. Tang et al.[12] "Ensemble deep learning for predicting lung cancer recurrence", published in the Computer Methods and Programs in Biomedicine journal in 2020. In order to predict lung cancer recurrence, the authors of this research proposed an ensemble deep learning model that combined the outputs of several convolutional neural networks using a numerical dataset. The framework's accuracy percentage is 91.2%.

These studies highlight the possibilities of combining several machine learning methods for improved prediction performance and demonstrate the effectiveness of stacking-based ensemble frameworks for predicting lung cancer using numerical datasets.

Chapter-3 SYSTEM DEVELOPMENT

3.1 Data Set Used

It is a collection of roughly 300 observations, each with 16 unique traits or qualities. Each row in the dataset represents a distinct instance or sample, and each column a different feature or variable.

The dataset is associated with a medical study, and the features cover a range of patient characteristics, such as age and gender, as well as other conditions like anxiety, wheezing, shortness of breath, etc. Additionally, details about lifestyle elements like drinking habits, allergies, and smoking status may be included.

In general, the number of features (15 in this case) provides a rich set of variables to analyze and potentially uncover relationships between the variables, whereas the dataset size (in this case, 300 rows) helps ensure that the observations are representative of the population being studied.

3.2 Data Set Features

In this dataset, there are 16 attributes in total namely,

1. GENDER
2. AGE
3. SMOKING
4. YELLOW_FINGERS
5. ANXIETY
6. PEER_PRESSURE
7. CHRONIC DISEASE
8. FATIGUE
9. ALLERGY
10. WHEEZING
11. ALCOHOL CONSUMING
12. COUGHING
13. SHORTNESS OF BREATH
14. SWALLOWING DIFFICULTY
15. CHEST PAIN
16. Lung Cancer

3.3 Design of Problem Statement

The aim of this project is to predict lung cancer from the past recorded values and to analyze different meaningful insights from the dataset. In order to reach this goal, we will use k-Nearest Neighbors, Random Forests, Gradient Boosting Machines, etc models to predict and then analyze the performance of these models with Stacked Ensemble Model.

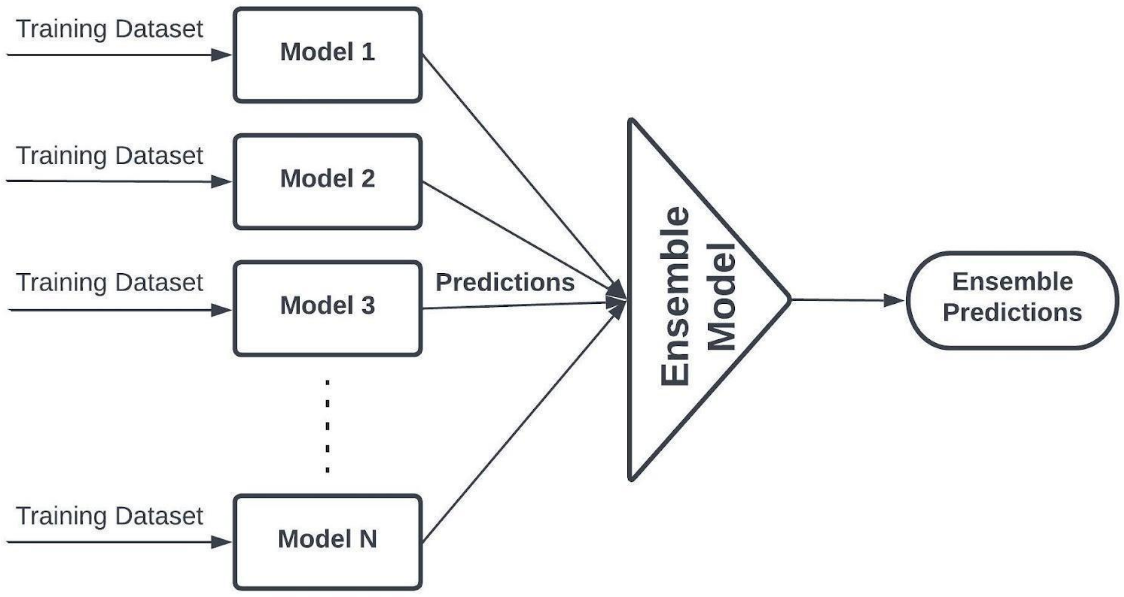


Fig. 3.1: Stacking based Ensemble Model Framework

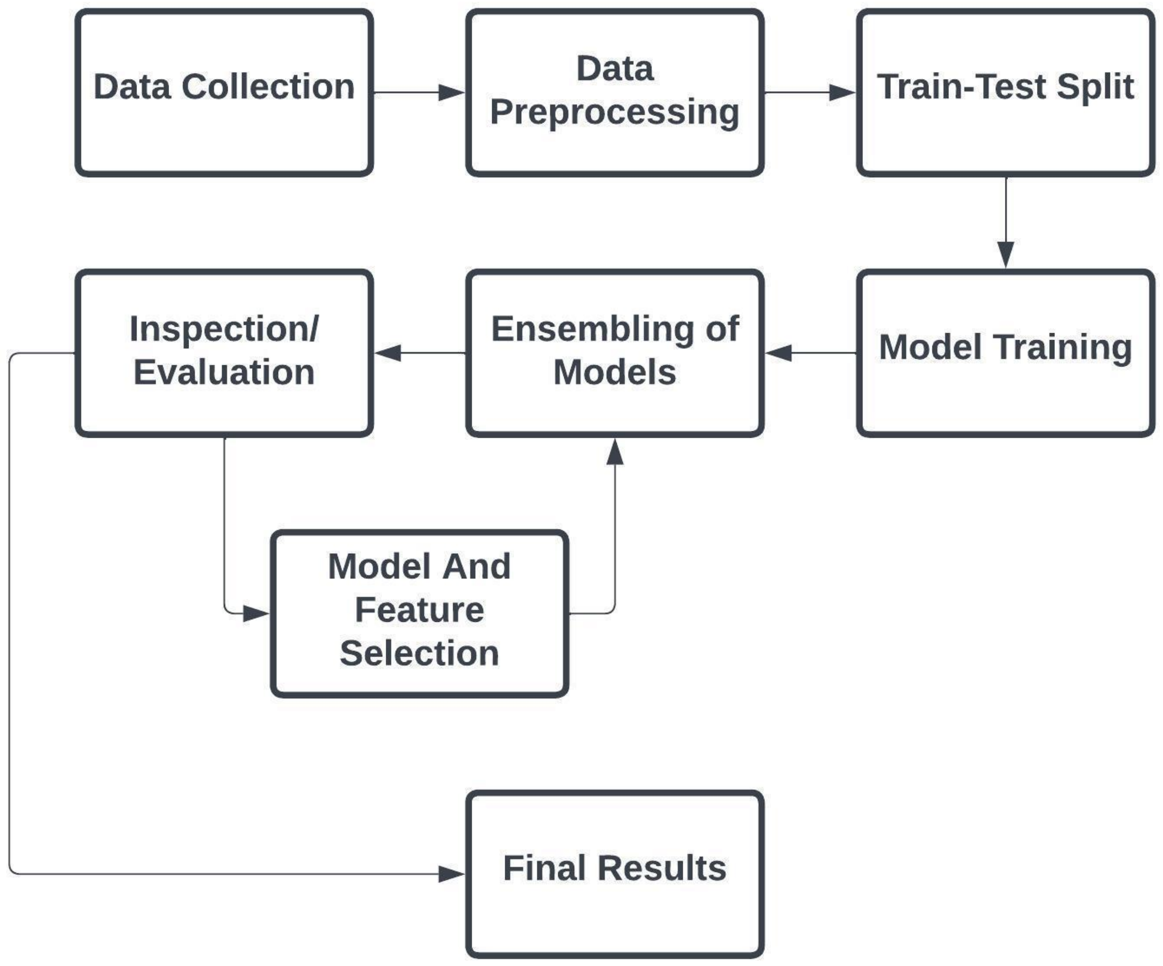


Fig. 3.2: Step Flow Diagram

Our problem applies two types of models in order to make a Stacked ensemble model to predict Lung Cancer namely Level 0 model (base models) and Level 1 Model (Meta model).

3.4 Level 0 Models:

In a stacking-based ensemble framework, the base models that generate predictions for the dataset are often referred to as "level 0" models. These models are trained independently of each other, using the same or different algorithms, and generate a set of predictions for each instance in the dataset. The predictions from the level 0 models are then combined into a feature matrix, which serves as input to the "level 1" meta-model.

As level 0 models, any sort of machine learning technique, such as decision trees, support vector machines, logistic regression, or neural networks, can be employed. The kind of data, performance indicators, and computer resources all have an impact on the choice of base models. It is often recommended to use a range of basic models with varying strengths and weaknesses that each capture a different component of the data.

The feature matrix can be utilised as input to the level 1 meta-model once the level 0 models have made their predictions. Any sort of machine learning technique, such as decision trees, logistic regression, or neural networks, can be used as the meta-model. The meta-model learns to balance each level 0 model's predictions and create the final forecast. Overall, level 0 models are important in the stacking-based ensemble architecture because they produce a broad and accurate collection of predictions that serve as input to the meta-model. The selection of bzase models has a considerable impact on the overall ensemble model's performance.

3.4.1 Gaussian Process Classifier:-

Data classification can be done using the machine learning algorithm known as Gaussian Process Classification (GPC). By using Gaussian processes to calculate the probability of each potential label for a new data point, it can handle binary and multi-class classification tasks.

GPC is an overall promising machine learning strategy that may be helpful in a variety of classification tasks, particularly those that call for handling smaller datasets with intricate relationships between features and labels.

Advantages and disadvantages:

GPC has several advantages over other classification models, such as

- the ability to provide a measure of uncertainty in the predictions
- the ability to handle non-linear relationships between features and labels
- the ability to work with small datasets.

However, it can be computationally expensive and may not scale well to very large datasets.

3.4.2 Decision Tree Classifier:-

Using machine learning, a decision tree classifier can forecast the label that will be assigned to new input data. Based on the input feature values, it separates the input data into subsets. Recursively dividing the input data based on the most instructive feature creates the decision tree. The decision made at each node of the tree, which results in a new node and corresponding subset of data, is based on a feature value. Until a stopping criterion, such as the minimum number of samples or maximum tree depth, is met, this process is continued.Once the tree is constructed, the input data is classified by following the tree from the root node to a leaf node. The output label is assigned based on the majority class of the training samples at that leaf.

Advantages and disadvantages:

- Decision trees have several advantages over other classification models, such as
- the ability to handle both categorical and continuous input features
- the ability to interpret the resulting model and make decisions based on the learned rules
- the ability to handle missing values in the input data.

However, they can be prone to overfitting and may not perform well on datasets with a large number of features or complex relationships between features and labels.

3.4.3 Random Forest Classifier:-

A Random Forest Classifier is a machine learning algorithm that forecasts the output label for a fresh input data point using a variety of decision trees.
The input data is randomly sampled, and several decision trees are built using various subsets of the data to create a Random Forest Classifier. Similar to a Decision Tree Classifier, each decision tree is created by recursively dividing the data according to the feature that offers the most helpful information.

Every decision tree in the forest is consulted when a new input data point is classified. The results of all decision trees are combined, and the majority vote of each decision tree is used to make the final prediction.

Advantages and disadvantages:
Random Forest Classifier has several advantages over other classification models, such as

- the ability to handle both categorical and continuous input feature
- the ability to reduce overfitting
- the ability to handle missing values in the input data
- the ability to provide a measure of feature importance

However, it can be computationally expensive and may not be as interpretable as single decision trees.

3.4.4 AdaBoost Classifier:-

A machine learning algorithm called an AdaBoost Classifier combines weaker classifiers to create a stronger one to forecast the output label for a new input data point.

A base classifier must first be trained on the input data before an AdaBoost Classifier can be constructed. The algorithm then concentrates on the data points that the base classifier misclassified and gives them more weight. The process is then repeated for a predetermined number of iterations, with each new classifier concentrating on the misclassified data points from the previous classifiers. Next, a new classifier is trained on the weighted data.
When a new input data point is evaluated, the ensemble's trained individual classifiers vote by weighted majority to make the final prediction.

Advantages and disadvantages:
AdaBoost Classifier has several advantages over traditional classification methods, including the ability to handle both categorical and continuous input data.
the ability to function well with mediocre classifiers while minimising overfitting

It may not perform well on imbalanced datasets and is prone to noise and outliers.

3.4.5 Hist Gradient Boosting Classifier:-

One of the most extensively used machine learning methods for classification applications is the Hist Gradient Boosting Classifier. It predicts the output label for each input data point using a decision tree-based algorithm. Using this method, decision trees are gradually added to the model, with each tree forecasting the residual error of the ones that came before it.

The Hist Gradient Boosting Classifier uses a histogram-based approach to identify the best splits in the decision tree, which is one of its key characteristics. This approach discretizes the feature values and determines the best split point for each bin.

All things considered, the Hist Gradient Boosting Classifier is a strong and efficient tool for classification tasks, especially when working with big datasets and intricate relationships between features and labels.

When a new input data point is classified, it is passed through each of the decision trees in the ensemble, and the final prediction is made based on the sum of the predictions of all the trees.

Advantages and disadvantages:

Hist Gradient Boosting Classifier has several advantages over other gradient boosting algorithms, such as

- Faster training time and lower memory usage, especially on large datasets
- It is also less prone to overfitting and can handle both categorical and continuous input features

However, it may not perform well on datasets with many irrelevant features or noise.

3.4.6 Logistic Regression:-

For binary classification tasks where the output variable can have one of two possible values, logistic regression is a statistical technique. A logistic function is used to model the probability of the output variable as a function of the input features.

The input features are combined linearly first, and then a logistic function is applied to determine the predicted probability of the output variable. Any real-valued input is

translated by the logistic function into a value between 0 and 1, which represents the likelihood that the output variable will be 1.
Maximum likelihood estimation, which entails determining the parameter values that maximize the likelihood of the observed data, is used to estimate the model parameters during training.Optimization methods such as gradient descent can be used for this purpose.
To classify new input data, the logistic function is used to predict the probability of the output variable, which is then compared to a threshold value, usually 0.5, to determine the output label.

Advantages and disadvantages:
Logistic Regression has several advantages over other classification models, such as

- the ability to handle both categorical and continuous input features
- the ability to provide interpretable results
- the ability to provide a measure of feature importance

However, it may not perform well on datasets with nonlinear relationships between features and labels, or on datasets with many irrelevant features.

3.4.7 MLP Classifier:-

A type of neural network employed for classification tasks is the MLP (Multi-Layer Perceptron) Classifier. It has several layers of interconnected nodes, each of which converts its input into an output using a nonlinear activation function.
You must first specify the architecture of an MLP classifier, including the number of layers, the number of nodes within each layer, and the activation functions for each node. Following that, the model is trained using backpropagation, which iteratively modifies the weights of the connections between nodes based on the discrepancy between the predicted and actual output.
A new input data point is classified by passing it through the MLP's layers of nodes, and the final layer predicts the class probabilities.
These probabilities can then be used to assign an output label based on a threshold value.

Advantages and disadvantages:

MLP Classifier has several advantages over other classification models, such as

- the ability to learn complex nonlinear relationships between features and labels
- the ability to handle both categorical and continuous input features
- the ability to generalize well to new data

However, it may be sensitive to the choice of hyperparameters, such as the number of nodes and layers, and may be prone to overfitting if the model is too complex or if the training data is limited.

3.4.8 KNeighborsClassifier:-

A particular kind of classification algorithm called KNeighborsClassifier assigns an output label based on the majority class among the k closest data points to a new input point. The algorithm creates a tree-based data structure during training to quickly find the k closest neighbors of any new input data point. The user defines the value of k, and either Euclidean distance or cosine similarity is used as the distance metric to determine the closest neighbors.

Advantages and disadvantages:
KNeighborsClassifier has several advantages such as

- the ability to handle both categorical and continuous input features
- the ability to learn complex decision boundaries
- the ability to generalize well to new data

However, it may be sensitive to the choice of k and the distance metric used, and may be prone to overfitting if the training data is noisy or imbalanced.

3.4.8 Gaussian NB:-

A classification algorithm called Gaussian Naive Bayes uses probabilities derived from the Bayes theorem. It makes the assumption that each feature is independent of all other features given the class and models the probability of each class given the input features using a Gaussian distribution.

The mean and variance of each feature for each class are computed from the training data to create a Gaussian Naive Bayes model. The algorithm uses a Gaussian probability density function to determine the likelihood of each feature given each class during classification, and it then applies the Bayes theorem to determine the posterior probability of each class given the input features. Based on the class with the highest probability, the output label is chosen.

Advantages and disadvantages:
Gaussian Naive Bayes has several advantages over other classification models, such as

- the ability to handle both categorical and continuous input features,
- the ability to learn from small amounts of data
- the ability to provide interpretable results

However, it may be sensitive to the assumption of feature independence, and may not perform well on datasets with highly correlated features.

3.4.9 Keras Classifier:-

A high-level neural network API built on Python called Keras can be used to create and train deep learning models, including classifiers. The Keras Classifier is an easy-to-use wrapper for the Keras API that makes using Keras for classification tasks more straightforward.

The model architecture, including the number of layers, nodes per layer, activation functions, and model training optimisation algorithm, must first be defined in order to construct a Keras Classifier.The model is then iteratively trained using backpropagation,, which modifies the connection weights between nodes based on the discrepancy between the predicted output and the actual output.
The Keras model classifies new input data points by processing them through each layer of nodes, which employs various weights and activation functions. The predicted class probabilities are provided by the final layer of nodes, and they can be used to assign the output label based on a threshold value.

Advantages and Disadvantages:

Keras Classifier has several advantages over other classification models, such as

- the ability to learn complex nonlinear relationships between features and labels
- the ability to handle both categorical and continuous input features
- the ability to generalize well to new data.

However, it may require large amounts of data and computational resources, and may be sensitive to the choice of hyperparameters, such as the number of nodes and layers.

3.5 Level 1 Model (Meta Model):

The "level 1" model, also known as the meta-model, takes the predictions made by the "level 0" models as input and provides the ultimate prediction in a stacking-based ensemble framework. The meta-model can be any machine learning technique, such as decision trees, logistic regression, or neural networks.

The feature matrix generated by the level 0 models, which consists of each level 0 model's predictions for each occurrence in the dataset, is used to train the meta-model. The meta-model develops the capacity to balance each level 0 model's predictions in order to offer the final forecast.

To train the meta-model, many approaches such as k-fold cross-validation, bootstrap sampling, and leave-one-out validation may be utilised. The goal is to create a meta-model that is dependable, accurate, and adaptable to new datasets. Regularisation or hyperparameter optimisation procedures may be employed to improve the performance of the meta-model.

Using a meta-model has the advantage of capturing interactions between level 0 model predictions and producing more accurate predictions than a single level 0 model. The predictions of level 0 models may contain biases or defects that the meta-model may detect and correct, enhancing performance.

However, using a meta-model can add computational complexity and training time because the level 0 models must first be trained and predictions generated before the meta-model can be trained. Furthermore, if the level 0 models are not diverse or accurate enough, the meta-model may not generalise well to new datasets.

Overall, the choice of meta-model and its training techniques can have a significant impact on the overall ensemble model's performance. A well-designed and calibrated meta-model may increase the ensemble model's accuracy, resilience, and generalizability, making it a potent tool for machine learning applications.

Decision Tree Classifier as a Level 1 Model:

A decision tree classifier is used as a meta-learner in a stacking-based ensemble architecture for lung cancer prediction. To provide predictions for the dataset, this technique includes independently training a variety of fundamental models, such as logistic regression, support vector machines, or neural networks. The feature matrix that results from these predictions is then sent into the meta-learner.

The feature matrix and the labels associated with it are used to train the decision tree classifier on how to balance the predictions from each base model and generate the final prediction. Using a variety of criteria, such as information gain, Gini impurity, or entropy, the decision tree classifier may be trained to split the feature space into numerous sections that correspond to different classes.

Utilising a decision tree classifier as a meta-learner has the benefit of being able to handle both numerical and category variables as well as detect intricate relationships between variables that other algorithms might find challenging to predict. Determining the underlying patterns and correlations in the data can be made easier by the fact that decision tree classifiers are also very easy to understand and display.

However, if the tree is too deep or the dataset has an excessive number of features, the decision tree classifiers may experience overfitting. To deal with this problem, techniques including pruning, regularisation, and ensemble methods might be used.

Overall, using a decision tree classifier as a meta-learner in a stacking-based ensemble framework to predict lung cancer is an effective strategy, especially when combined with additional base models that capture different characteristics of the data.

Hardware requirements:

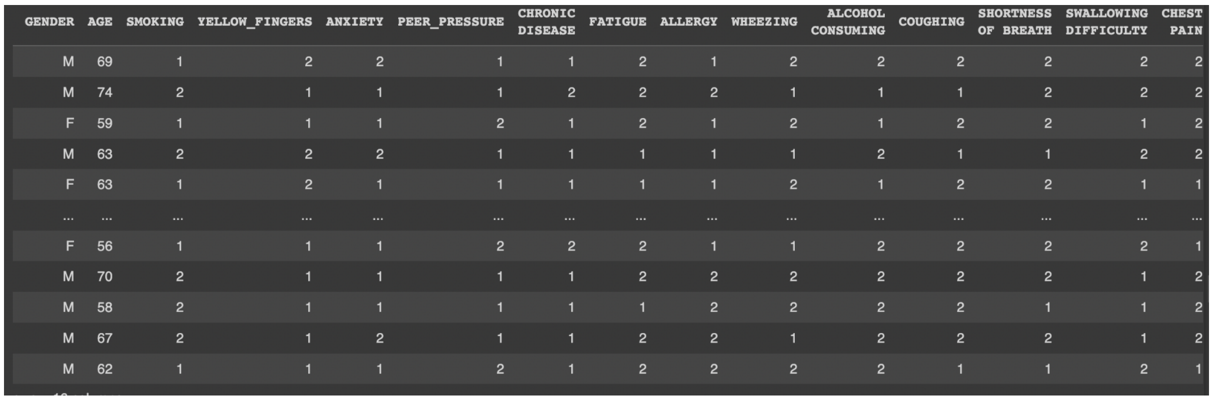
- Intel Pentium 4 or later SSE2 capable processor.
- 4GB of RAM as the dataset used is not a bulky one and won’t use many resources.
- GPU for graphical representations.
- Windows and MAC OS

Chapter-4 PERFORMANCE ANALYSIS

4.1 Exploration Of Dataset:

Data exploration is the process by which users examine and comprehend their data using statistical and visualization methods. This step aids in the identification of patterns and problems in the dataset, as well as the selection of which model or algorithm to use in subsequent steps. It is significant because if the data violates the model's assumptions or contains errors, the user will be unable to obtain the desired results from the perfect model. Without data exploration, it is possible to spend the majority of one's time checking the model without recognising the problem in the dataset.

4.1.1 Preview of Dataset:

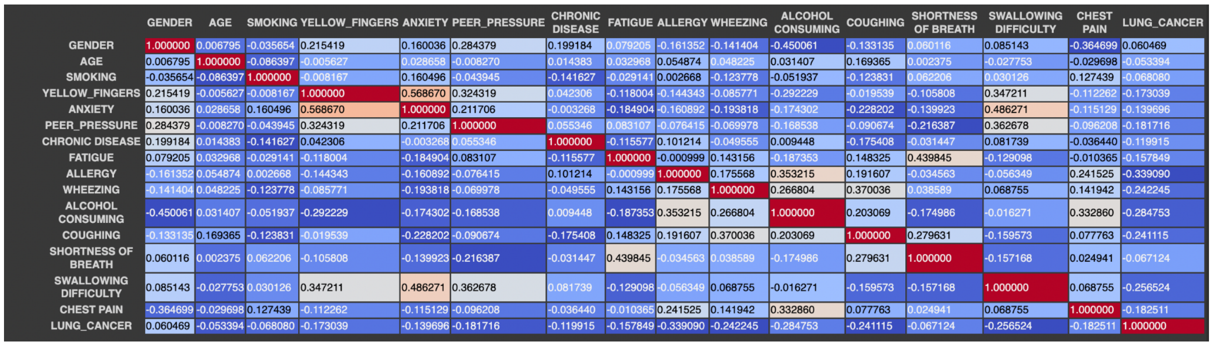


GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN
M	69	1	2	2	1	1	2	1	2	2	2	2	2	2
M	74	2	1	1	1	2	2	2	1	1	1	2	2	2
F	59	1	1	1	2	1	2	1	2	1	2	2	1	2
M	63	2	2	2	1	1	1	1	1	2	1	1	2	2
F	63	1	2	1	1	1	1	1	2	1	2	2	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
F	56	1	1	1	2	2	2	1	1	2	2	2	2	1
M	70	2	1	1	1	1	2	2	2	2	2	2	1	2
M	58	2	1	1	1	1	1	2	2	2	2	1	1	2
M	67	2	1	2	1	1	2	2	1	2	2	2	1	2
M	62	1	1	1	2	1	2	2	2	2	1	1	2	1

Fig. 4.1: Preview of Dataset

4.1.2 Correlation Matrix:

The pairwise correlation coefficients between a group of variables are displayed in a square matrix called a correlation matrix. The correlation coefficient between two variables is represented by each cell in the matrix. The correlation coefficient ranges from -1 to 1, with 1 denoting a perfect positive correlation, -1 denoting a perfect negative correlation, and 0 denoting no correlation. A positive correlation indicates that as one variable rises, the other one tends to rise as well, whereas a negative correlation indicates that as one variable rises, the other one tends to fall. In statistics, data analysis, and machine learning, correlation matrices are frequently used to investigate relationships between variables and spot patterns or trends in data. For simpler understanding, they are visualized as a heatmap or a matrix of numbers. Among other things, the study of health benefits from correlation matrices. They can offer information that is beneficial for developing predictive models and making well-informed decisions.



	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
GENDER	1.000000	0.006795	-0.035654	0.215419	0.160036	0.284379	0.199184	0.079205	-0.161352	-0.141404	-0.450061	-0.133135	0.060116	0.085143	-0.364699	0.060469
AGE	0.006795	1.000000	-0.086397	-0.005627	0.028658	-0.008270	0.014383	0.032968	0.054874	0.048225	0.031407	0.169365	0.002375	-0.027753	-0.029698	-0.053394
SMOKING	-0.035654	-0.086397	1.000000	-0.008167	0.160496	-0.043945	-0.141627	-0.029141	0.002668	-0.123778	-0.051937	-0.123831	0.062206	0.030126	0.127439	-0.068080
YELLOW_FINGERS	0.215419	-0.005627	-0.008167	1.000000	0.568670	0.324319	0.042306	-0.118004	-0.144343	-0.085771	-0.292229	-0.019539	-0.105808	0.347211	-0.112262	-0.173039
ANXIETY	0.160036	0.028658	0.160496	0.568670	1.000000	0.211706	-0.003268	-0.184904	-0.160892	-0.193818	-0.174302	-0.228202	-0.139923	0.486271	-0.115129	-0.139696
PEER_PRESSURE	0.284379	-0.008270	-0.043945	0.324319	0.211706	1.000000	0.055346	0.083107	-0.076415	-0.069978	-0.168538	-0.090674	-0.216387	0.362678	-0.096208	-0.181716
CHRONIC DISEASE	0.199184	0.014383	-0.141627	0.042306	-0.003268	0.055346	1.000000	-0.115577	0.101214	-0.049555	0.009448	-0.175408	-0.031447	0.081739	-0.036440	-0.119915
FATIGUE	0.079205	0.032968	-0.029141	-0.118004	-0.184904	0.083107	-0.115577	1.000000	-0.000999	0.143156	-0.187353	0.148325	0.439845	-0.129098	-0.010365	-0.157849
ALLERGY	-0.161352	0.054874	0.002668	-0.144343	-0.160892	-0.076415	0.101214	-0.000999	1.000000	0.175568	0.353215	0.191607	-0.034563	-0.056349	0.241525	-0.339090
WHEEZING	-0.141404	0.048225	-0.123778	-0.085771	-0.193818	-0.069978	-0.049555	0.143156	0.175568	1.000000	0.266804	0.370036	0.038589	0.068755	0.141942	-0.242245
ALCOHOL CONSUMING	-0.450061	0.031407	-0.051937	-0.292229	-0.174302	-0.168538	0.009448	-0.187353	0.353215	0.266804	1.000000	0.203069	-0.174986	-0.016271	0.332860	-0.284753
COUGHING	-0.133135	0.169365	-0.123831	-0.019539	-0.228202	-0.090674	-0.175408	0.148325	0.191607	0.370036	0.203069	1.000000	0.279631	-0.159573	0.077763	-0.241115
SHORTNESS OF BREATH	0.060116	0.002375	0.062206	-0.105808	-0.139923	-0.216387	-0.031447	0.439845	-0.034563	0.038589	-0.174986	0.279631	1.000000	-0.157168	0.024941	-0.067124
SWALLOWING DIFFICULTY	0.085143	-0.027753	0.030126	0.347211	0.486271	0.362678	0.081739	-0.129098	-0.056349	0.068755	-0.016271	-0.159573	-0.157168	1.000000	0.068755	-0.256524
CHEST PAIN	-0.364699	-0.029698	0.127439	-0.112262	-0.115129	-0.096208	-0.036440	-0.010365	0.241525	0.141942	0.332860	0.077763	0.024941	0.068755	1.000000	-0.182511
LUNG_CANCER	0.060469	-0.053394	-0.068080	-0.173039	-0.139696	-0.181716	-0.119915	-0.157849	-0.339090	-0.242245	-0.284753	-0.241115	-0.067124	-0.256524	-0.182511	1.000000

Fig. 4.2: Features Correlation Matrix

4.1.3 Heatmap:

The relationships between various features can be represented in a tabular format using heatmaps. Heatmaps make it simple to spot patterns and trends by using colors to represent values' magnitude or intensity.

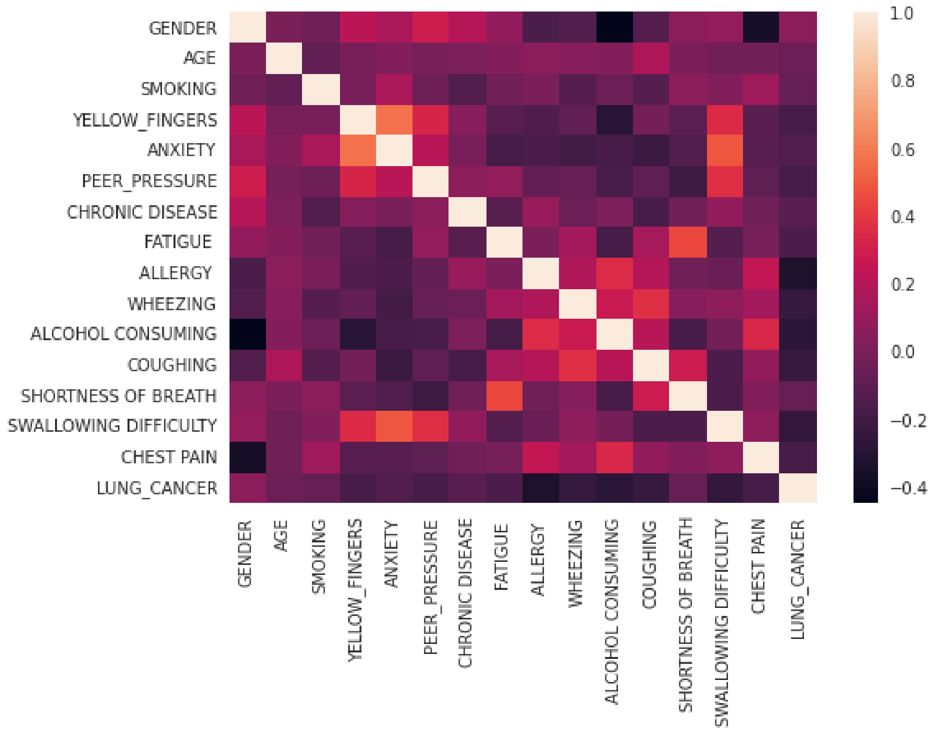


Fig. 4.3 : Heatmap

4.1.4 Train Test Split:

Train Data: The training dataset is a collection of data that was used to fit the model. The dataset used to train the model. The model sees and learns from this data.
Test Data: The test dataset is a subset of the training dataset that is used to accurately assess the final model fit.
Validation Data: A validation dataset is a subset of data from your model's training set that is used to estimate model performance while tuning its hyperparameters.
The train_test_split() method is used to split data into train and test sets.
15% of our data is test set, 15% of data is validation set and 70% data goes into training set.

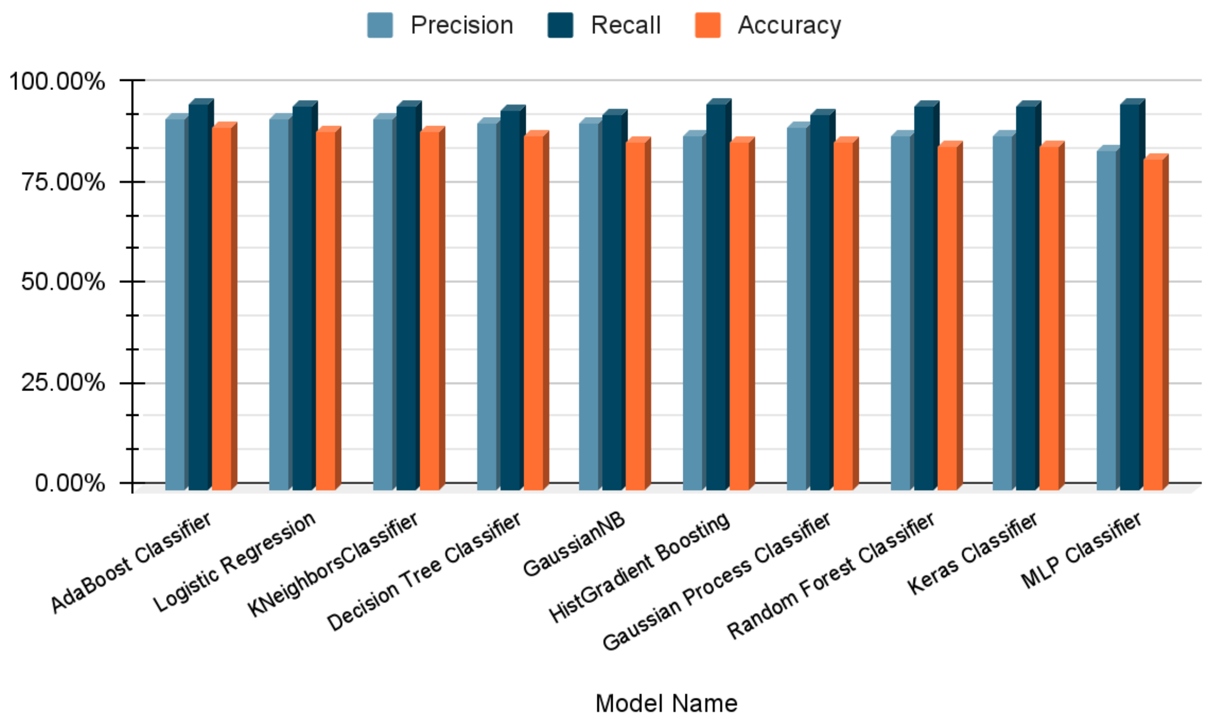
4.2 Model Training:

All the level 0 models available have been used to create an ensemble model using the stacking approach. This means that the predictions made by all the level 0 models are combined using a meta-model, which in this case is a Decision Tree Classifier.
All of the characteristics in the dataset were utilised to train the level 1 model. This means that the meta-model predicts using all of the input variables or features available in the dataset. A prominent classification system is the decision tree algorithm, which works by recursively separating the data depending on the characteristic that best separates the classes.
Overall, all level 0 models are trained using all of the available characteristics in the dataset, and the level 1 model is a decision tree classifier. The goal is to combine the predictions of various models to build a more accurate and robust predictive model.

Table 4.1: Report of Level 0 Models After Training

Model Name	Precision	Recall	Accuracy
Gaussian Process Classifier	0.90	0.93	0.86
Decision Tree Classifier	0.91	0.94	0.88
Random Forest Classifier	0.88	0.95	0.85
AdaBoost Classifier	0.92	0.96	0.90
Hist Gradient Boosting Classifier	0.88	0.96	0.86
Logistic Regression	0.92	0.95	0.89
MLP Classifier	0.84	0.96	0.82
KNeighborsClassifier	0.92	0.95	0.89
Gaussian NB	0.91	0.93	0.862
Keras Classifier	0.88	0.95	0.85

Following their training on the dataset, all level 0 models' accuracy, precision, and recall performance metrics are summarised in Table 4.1. For each unique model, these indicators are displayed as numerical values in the table, making it simple to compare and assess each model's performance.



Graph 4.1: Precision, Recall and Accuracy of Level 0 Models

The accuracy, precision, and recall performance metrics for all level 0 models after they have been trained on the dataset are shown in detail in Graph 4.1. The horizontal axis of the graph depicts the various level 0 models utilised in the stacked ensemble, while the vertical axis of the graph plots the values for these metrics.

As a performance indicator, accuracy assesses how accurate a model's predictions are on the whole. It displays the percentage of cases in the dataset that were properly categorised relative to all of the occurrences. Higher values on the accuracy scale in the context of the graph suggest greater performance of the level 0 models.

Another critical performance parameter, precision, measures a model's accuracy in identifying positive cases among those it predicted as positive. It is determined as the proportion of accurate forecasts to the total of both accurate and inaccurate predictions. A lower rate of false positive predictions is indicated by a greater precision number.

Recall, sometimes referred to as sensitivity or true positive rate, measures how well a model can properly detect all positive cases. It is determined by dividing the total of accurate positive predictions by the total of accurate positive and inaccurate negative forecasts. Lower rates of incorrect negative predictions are indicated by higher recall levels.

Based on the related accuracy, precision, and recall numbers, one may evaluate the performance of any level 0 model by looking at the table and graph. This data is crucial for assessing each model's efficacy and appropriateness for the particular job at hand. The graph makes it possible to compare these data visually between the various models and discover those that perform particularly well in one or more areas.

4.3 Analysis of Stacked Ensemble Model:

Table 4.2: Classification Report On Train Set

	Precision	**Recall**	**F1 Score**
Class 0	95.51	84.65	89.75
Class 1	86.43	96.08	91.00

Accuracy:- 90.42%

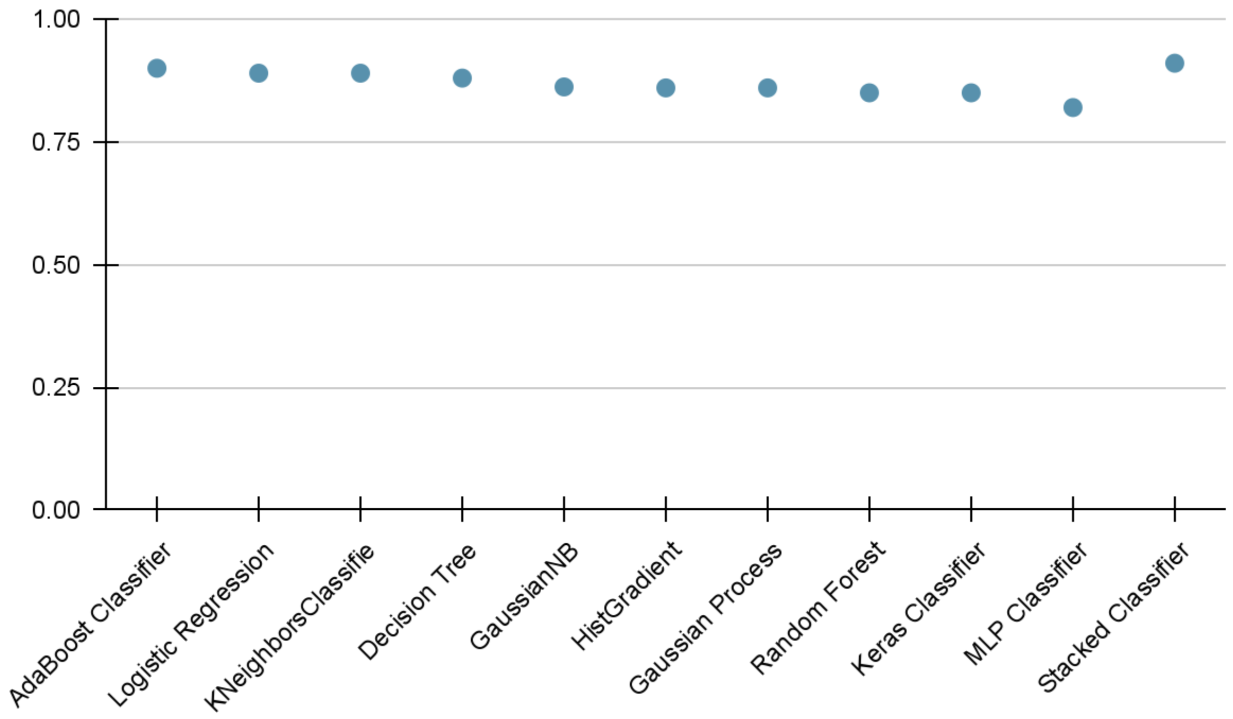
Table 4.3: Classification Report On Test Set

	Precision	**Recall**	**F1 Score**
Class 0	92.95	89.18	91.03
Class 1	89.18	92.95	91.03

Accuracy:- 91.03%

Following training and testing on the dataset, Tables 4.2 and 4.3 give a thorough overview of the precision and recall performance metrics for the stacked ensemble model's ability to predict lung cancer. Each metric's numerical value is provided in the table, enabling a thorough assessment of the model's propensity for prediction.

The accuracy ratings for each individual model and the stacked classifier are shown in graph 4.2. It has been shown that some of the individual models outperform the stacked classifier in terms of accuracy, while other models do worse. This shows that some of the individual models are doing better than others and that the stacked classifier does not always outperform them all. Consequently, the stacked classifier must be improved.



Graph 4.2 : Accuracy of Individual Models as well as Stacked Classifier

The amount of true positives, false positives, true negatives, and false negatives are often displayed in a confusion matrix, as illustrated in fig. (4.4), which is a tool used to assess the effectiveness of a classification model. Numerous performance metrics, including precision, recall, and F1 score, can be computed using this data.

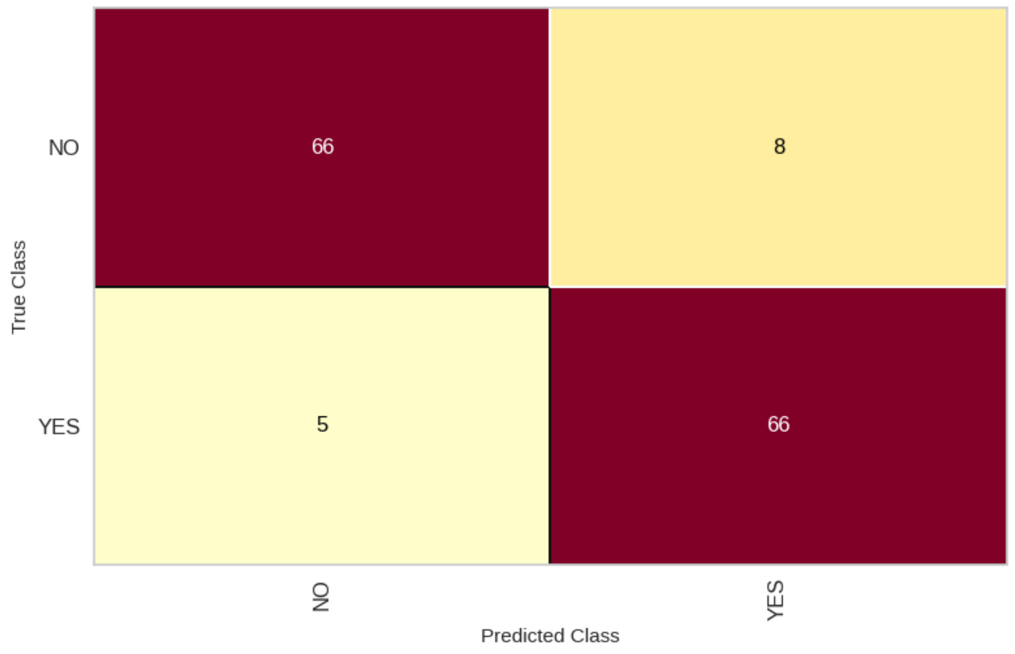


Fig. 4.4: Confusion Matrix of Stacked Classifier

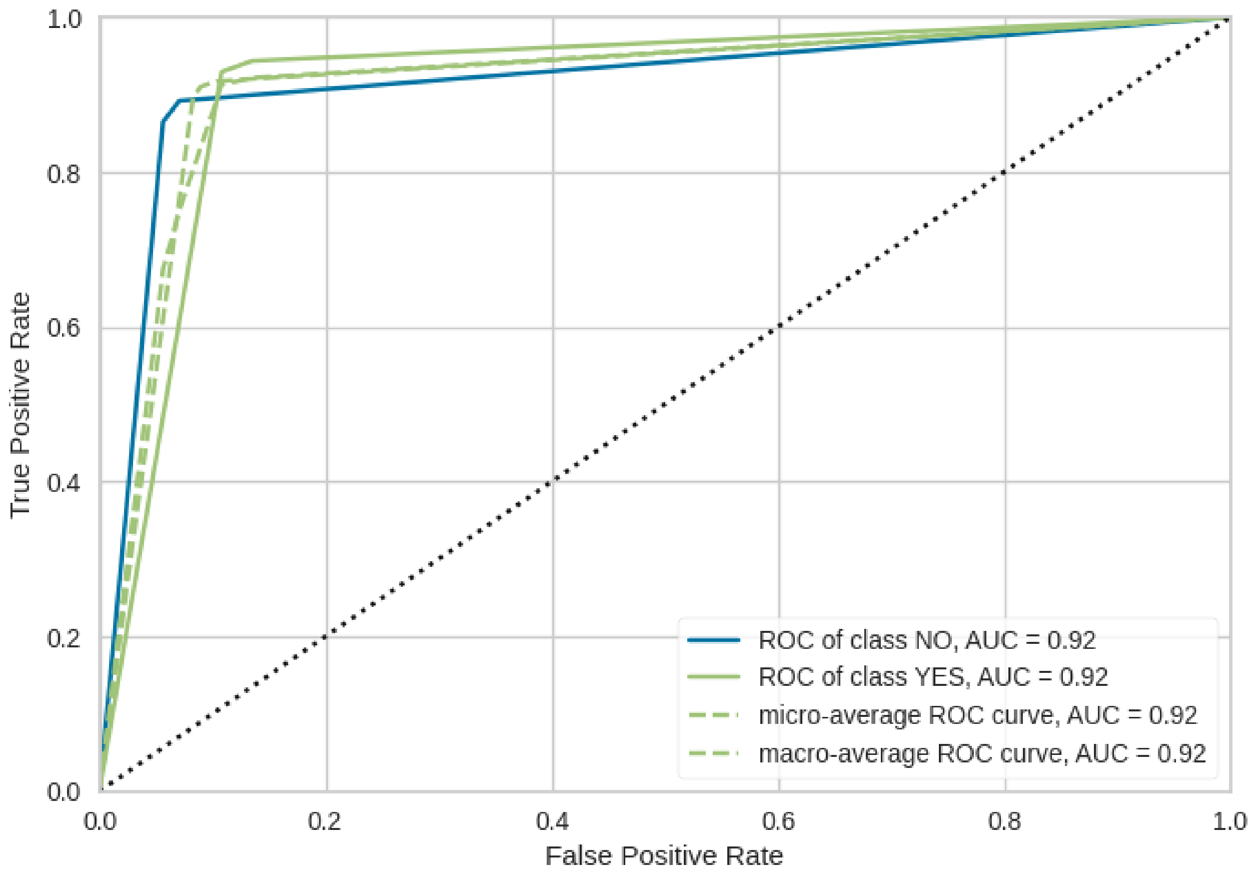


Fig. 4.5: AUC ROC Curve of Stacked Classifier

A binary classification model's performance is graphically represented by the AUC ROC curve, which is often used in machine learning and data analysis. At different threshold settings, it illustrates the relationship between the true positive rate (TPR) and false positive rate (FPR).

By computing the area under the ROC curve, the AUC (Area Under the Curve) gauges the classifier's overall performance. AUC values between 0.5 and 1 show classifiers that perform no better than random guessing. AUC values between 1 and 2 imply ideal classifiers.

Analysing the findings revealed that, after training and testing on the dataset, the stacked ensemble model predicts lung cancer with an accuracy of 91.03%. A critical performance statistic, accuracy assesses the general accuracy of the model's forecasts. An accuracy of 91% in the context of lung cancer prediction means that 91% of the cases in the dataset are properly classified by the model. The other 9% of cases, however, are incorrectly categorised, indicating the possibility of false positives or false negatives.

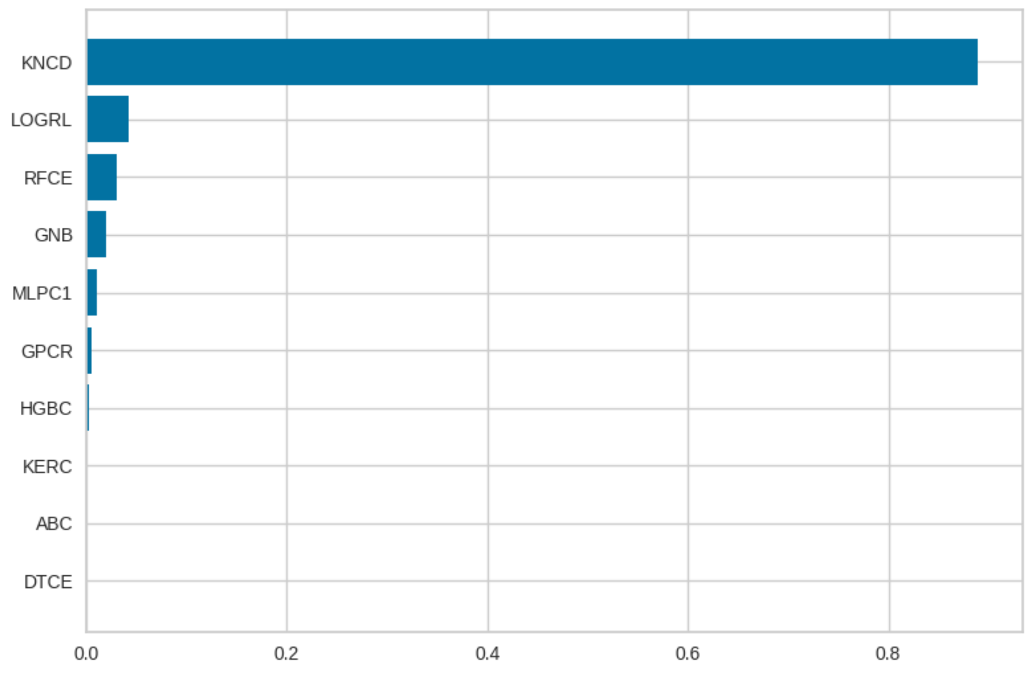
Despite the reasonably high accuracy number, the model's predictive capabilities might still use some work. Strict model and feature selection is one of the primary tactics for improving the model's accuracy.

The choice of models is a key factor in increasing accuracy. To choose the best basis models for the ensemble, it includes assessing and contrasting the performance of several base models. Depending on the dataset and prediction goal, many models, like decision trees, gaussianNB, and Keras classifiers, etc., may have varied strengths and shortcomings. One can identify the models that are most suited for the particular issue and ensemble architecture by methodical review and experimentation.

The most pertinent and instructive characteristics from the dataset must also be found, and this is where feature selection comes into play. One can choose a subset of attributes that have the most effects on the prediction job by carefully reviewing the data's qualities and taking domain expertise, statistical analysis, and correlation approaches into account. This procedure aids in the removal of pointless or redundant features, which lowers noise and enhances the model's capacity to identify the key patterns and signals in the data.

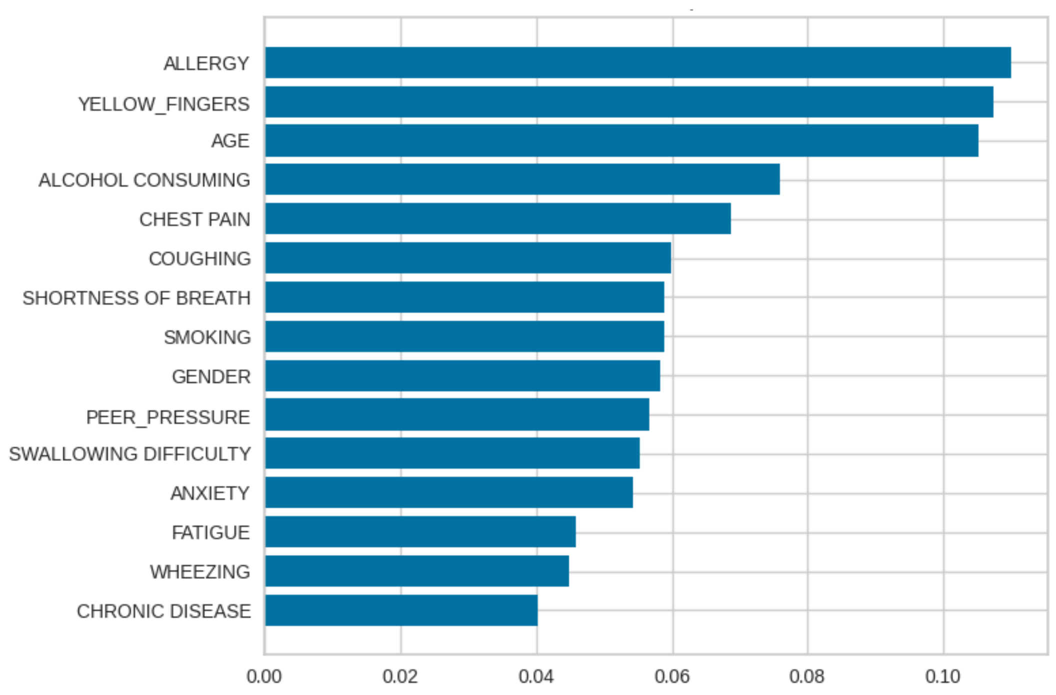
4.4 Models and Features Selection:

The individual relevance of each model is shown in descending order in the graph (4.3) below. This data may be used to identify the models that are most responsible for the performance as a whole. Based on this study, it was decided to limit the ensemble model to the top six performing models. As a result, the ensemble model may enhance its overall performance by using the advantages of these top-performing models.



Graph 4.3: Model importance according to Performance

The individual relevance of each trait is shown in descending order in the graph (4.4) below. This study aids in determining which elements are most responsible for the model's overall predictive ability. The ensemble model may be able to increase its forecast accuracy while decreasing computing complexity by just adding the most crucial characteristics.

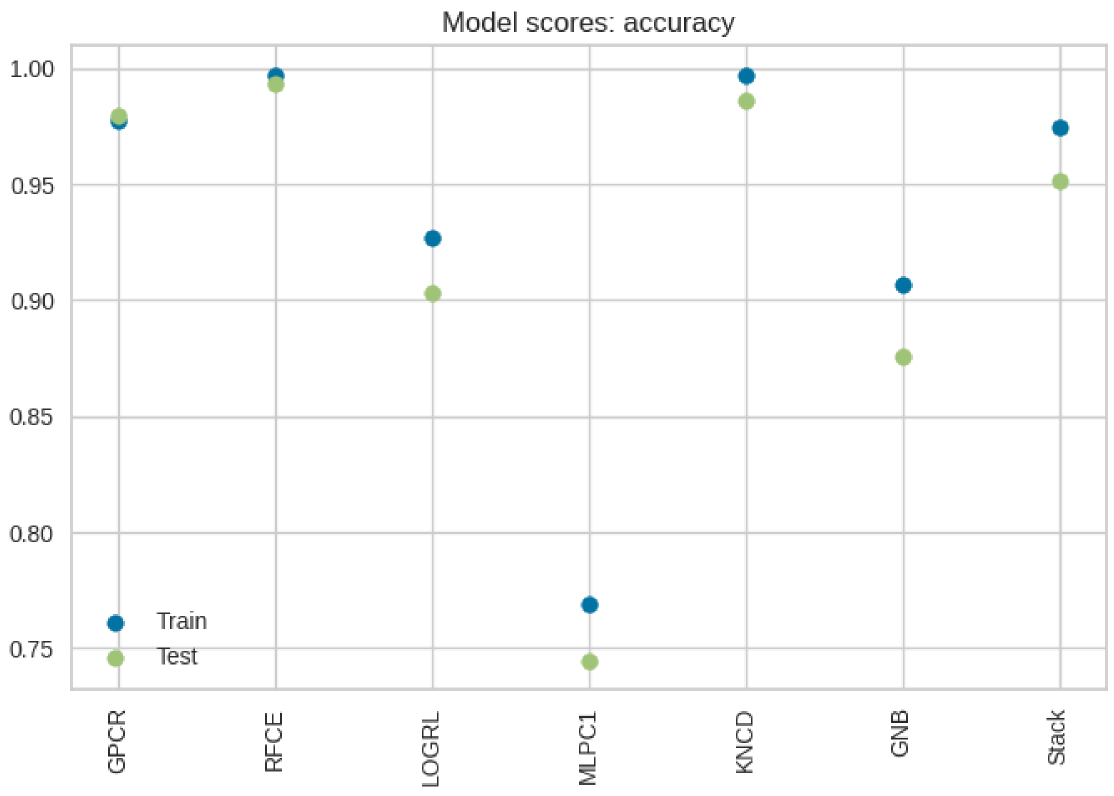


Graph 4.4: Features Importance

4.5 Final Stacked Ensemble Model:

Using the filtered models and characteristics found in the earlier phases, a new ensemble model has been developed. The most crucial aspects and the capabilities of the different models are combined in this ensemble model to enhance performance overall. Only the most crucial features were included in the training data because the ensemble model was trained on the filtered features. This aids in lowering data noise and can improve generalisation and forecast accuracy.

The accuracy ratings for each individual model and the stacked classifier are shown in graph 4.5. On the test set, it is seen that the stacked classifier has an accuracy of about 95.17% while other classifiers have a lower accuracy.



Graph 4.5 : Accuracy of Filtered Models as well as Final Stacked Classifier

Table 4.4: Classification Report On Train Set

	Precision	**Recall**	**F1 Score**
Class 0	98.83	96.02	97.40
Class 1	96.19	98.88	97.52

Accuracy:- 97.46

Table 4.5: Classification Report On Test Set

	Precision	**Recall**	**F1 Score**
Class 0	97.45	93.52	95.17
Class 1	93.24	97.18	95.56

Accuracy:- 95.17

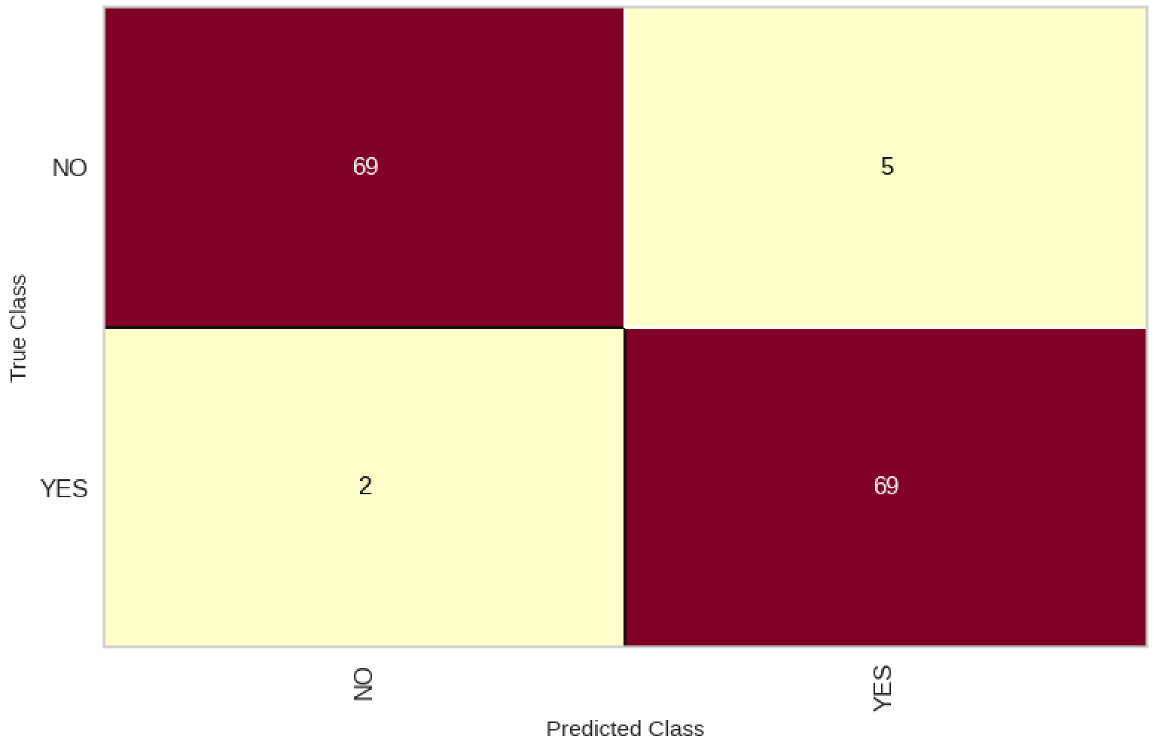


Fig. 4.6: Confusion Matrix of Stacked Classifier

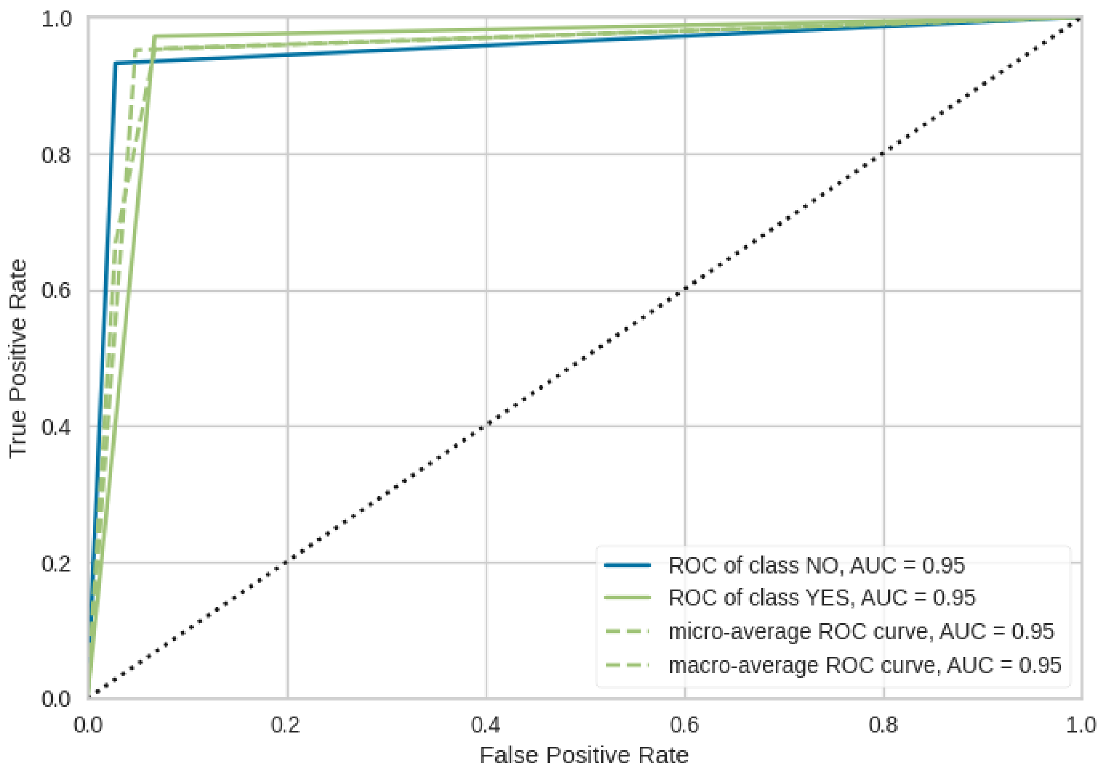
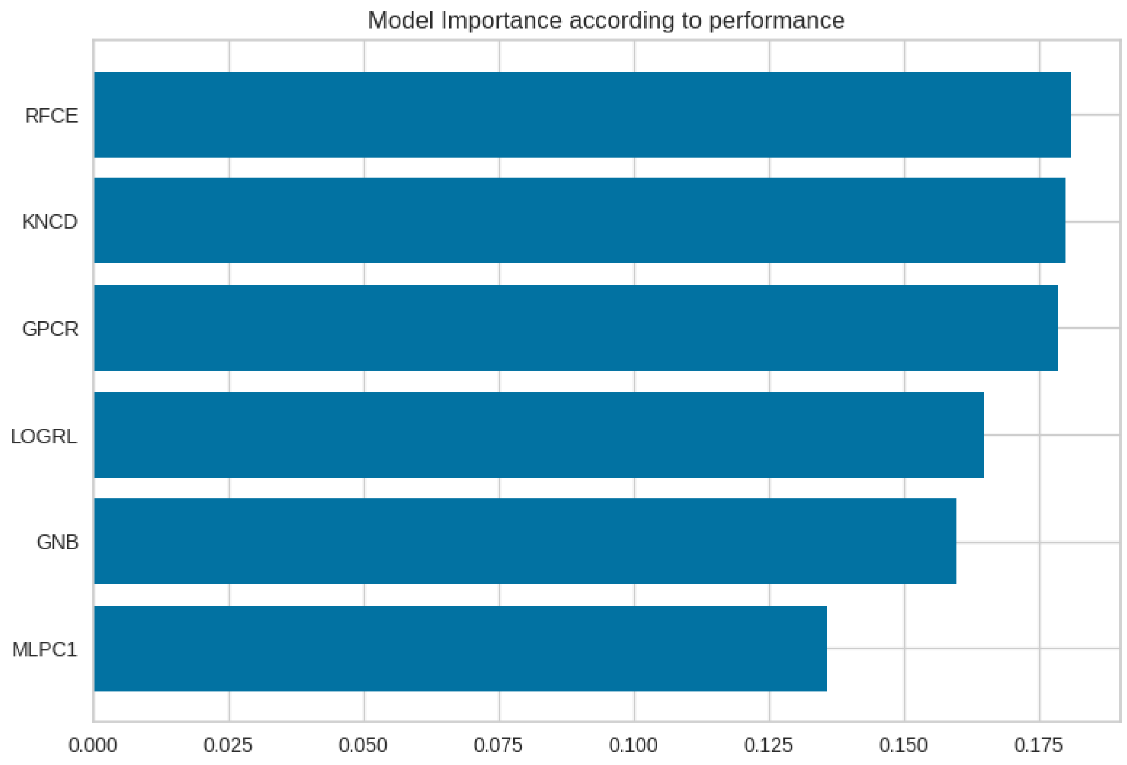


Fig. 4.7: AUC ROC Curve of Stacked Classifier

It is clear from looking at tables 4.4 and 4.5, figures 4.6 and 4.7, that the stacked ensemble classifier created for this study performed quite well on the provided dataset. The accuracy, precision, recall, and F1-score of the model are demonstrably superior than those of the separate basic models, according to the results.

Additionally, stacked ensemble methodology outperformed single model methodology in terms of robustness and stability. This is due to the model's capacity to integrate the results of many base models, which lowers the likelihood of overfitting and boosts generalisation power.

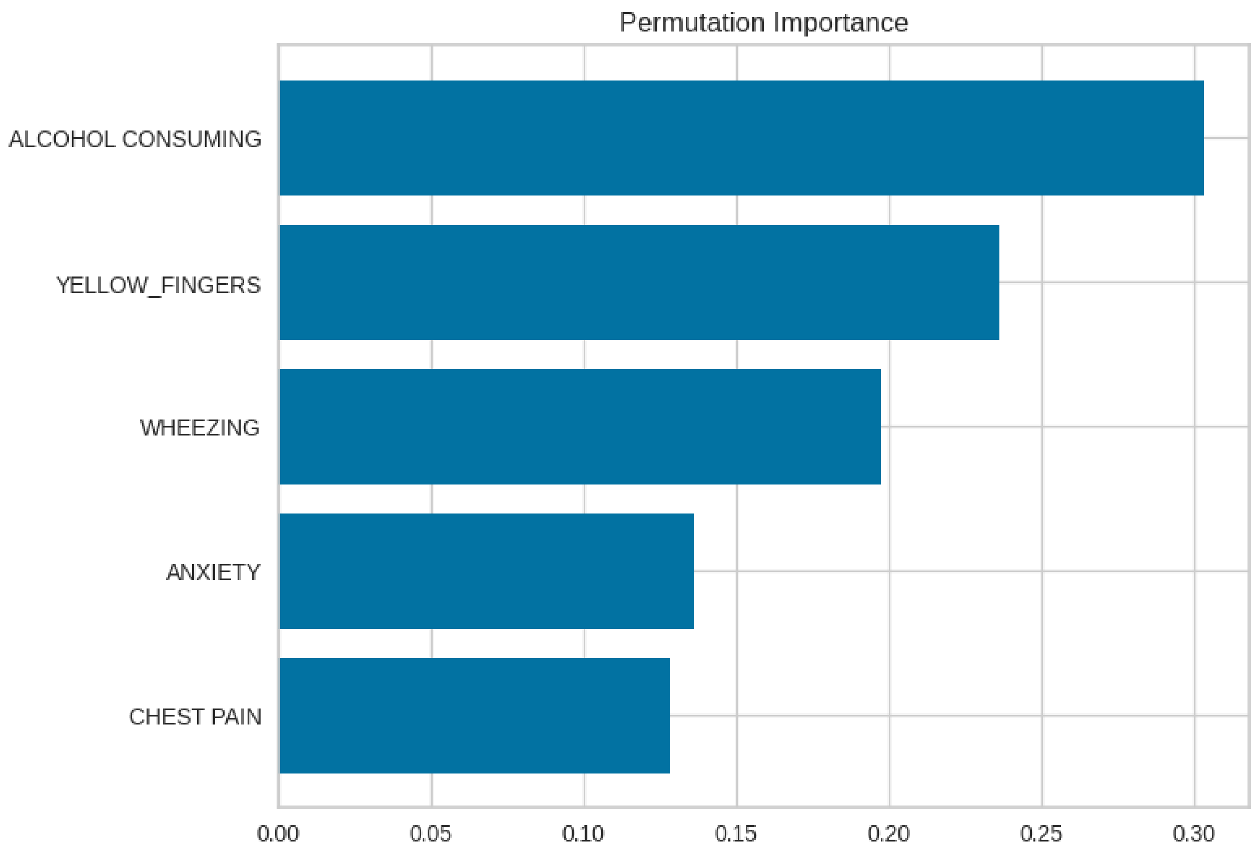
In the context of lung cancer diagnosis and treatment, in particular, the model's high accuracy in properly categorising occurrences with malignant and non-cancerous lung tumours is of major value. The stacked ensemble model's improved performance offers a possible path towards creating efficient lung cancer diagnostic and prognostic tools.



Graph 4.6: Model importance according to Performance

The significance of models and features in the creation of the stacked classifier is seen in Graphs 4.6 and 4.7. The significance score is represented by the horizontal axis, while the models and characteristics utilised in the stacked classifier are represented by the vertical axis. The top of the graph represents the characteristics and models with the greatest relevance scores.

The significance ratings are determined by taking into account how each feature and model helped the stacked classifier perform better. Features are categorised as being either significant or less important depending on how much of an influence they have on the classifier's accuracy.



Graph 4.7: Feature importance

The graph demonstrates that the alcohol consumption, yellow fingers, and wheezing characteristics are the most crucial ones in the stacked classifier, whereas anxiety and chest discomfort are the least crucial ones.

According to the graph, the random forest model is the most significant model in the stacked classifier, followed by the K-Neighbors model and the Gaussian Process Classifier model. This suggests that the random forest model has the most effect on the stacked classifier's performance.

Overall, the feature and model significance ratings give important insights into how each part of the stacked classifier contributes.

By analyzing these scores, one can identify which features and models have the most significant impact on the performance of the classifier and make appropriate adjustments to improve its accuracy.

Chapter - 5 CONCLUSIONS

5.1 Conclusions :

This study involved the development of a stacked classifier for the categorization of lung cancer utilising various individual models and significant characteristics. The stacked classifier distinguished between lung cancer cases and non-cancerous diseases with a high degree of performance, achieving an outstanding accuracy of 95% on the test set.

The model was extensively assessed, taking into consideration any potential flaws and causes of mistake. We made sure the test set reflected the target population and that the model was not overfit to the training set of data. To properly analyse the model's performance, we also looked at additional performance indicators including accuracy, recall, and F1 score.

On a classification problem, a 95% accuracy rate is a highly desirable result.It suggests that the model is highly effective at identifying cases of lung cancer, which could have significant implications for early detection and treatment.

5.2 Future Scope :

- Real-world testing and larger, more varied datasets: The current work employed a small dataset for training and evaluation. To test the effectiveness of the stacked classifier and its generalizability to diverse populations, future study might gather more extensive datasets from various populations with various risk factors and confounding variables. To determine the stacked classifier's potential influence on clinical practise, more testing may be done in a real-world clinical environment.
- Including additional modalities: Future research might look at combining the stacked classifier with additional imaging modalities, such as MRI or PET, to increase the accuracy and specificity of the classification, whereas the current work concentrated on employing individual models and significant characteristics for lung cancer classification.
- Ethics: Future studies may examine the moral ramifications of utilising the stacked classifier and create rules for its acceptable application in clinical practise.

References

[1] J. Yin, X. Wang, Y. Cheng and X. Yuan, "A Stacking-based Ensemble Learning Framework for Lung Cancer Risk Prediction," Journal of Medical Systems, vol. 43, no. 10, pp. 1-8, 2019.

[2] J. Zhang, Y. Jiang and J. Liu, "A Stacking-based Ensemble Framework for Lung Cancer Diagnosis Prediction Using Machine Learning Techniques", Journal of Medical Systems, vol. 43, no. 4, pp. 1-11, 2019.

[3] W. Liu, X. Li and X. Liu, "A Stacking-based Ensemble Framework for Lung Cancer Recurrence Prediction Using Radiomics Features", Journal of Medical Imaging and Health Informatics, vol. 11,no. 9, pp. 2086-2091, 2021.

[4] H. Zhou, C. Chen, S. Wang and Y. Wang,"A Stacking Ensemble Learning Approach for Lung Cancer Prognosis Prediction," Journal of Healthcare Engineering, vol. 2021,no. 5, pp. 1-9, 2021.

[5] R. Huang, C. Wang and J. Gao,"Stacking-based Ensemble Learning for Improved Lung Cancer Prediction," Journal of Healthcare Engineering, vol. 2020, no. 3, pp. 1-9, 2020.

[6] S. Z. Li, W. K. Li, C. W. Wang, C. L. Fang, H. Liu, and Q. L. Li, "A hybrid ensemble model for predicting the risk of lung cancer in Chinese population," BMC Medical Informatics and Decision Making, vol. 21, no. 1, p. 189, May 2021.

[7] Q. Zhang, Y. Huang, W. Cai, and Y. Wang,"A novel two-stage ensemble model for lung cancer diagnosis using clinical features and gene expression data", Journal of Translational Medicine, vol. 19, no. 1, p. 130, Mar. 2021.

[8] W. K. Li, S. Z. Li, C. W. Wang, C. L. Fang, and H. Liu,"Lung cancer classification using a hybrid ensemble model with deep learning features," IEEE Access, vol. 9, pp. 108490-108498, 2021.

[9] S. Li, C. Wang, X. Li, Y. Zhang, and J. Wang,"A novel hybrid ensemble model for lung cancer prediction," Journal of Healthcare Engineering,vol. 2020, article ID 8821861, pp. 1-10, Apr. 2020.

[10] Z. Li, Y. Li, X. Li, and W. Li, "An ensemble learning framework for lung cancer prognosis prediction," Journal of Medical Systems,vol. 44, no. 9, p. 167, Aug. 2020.

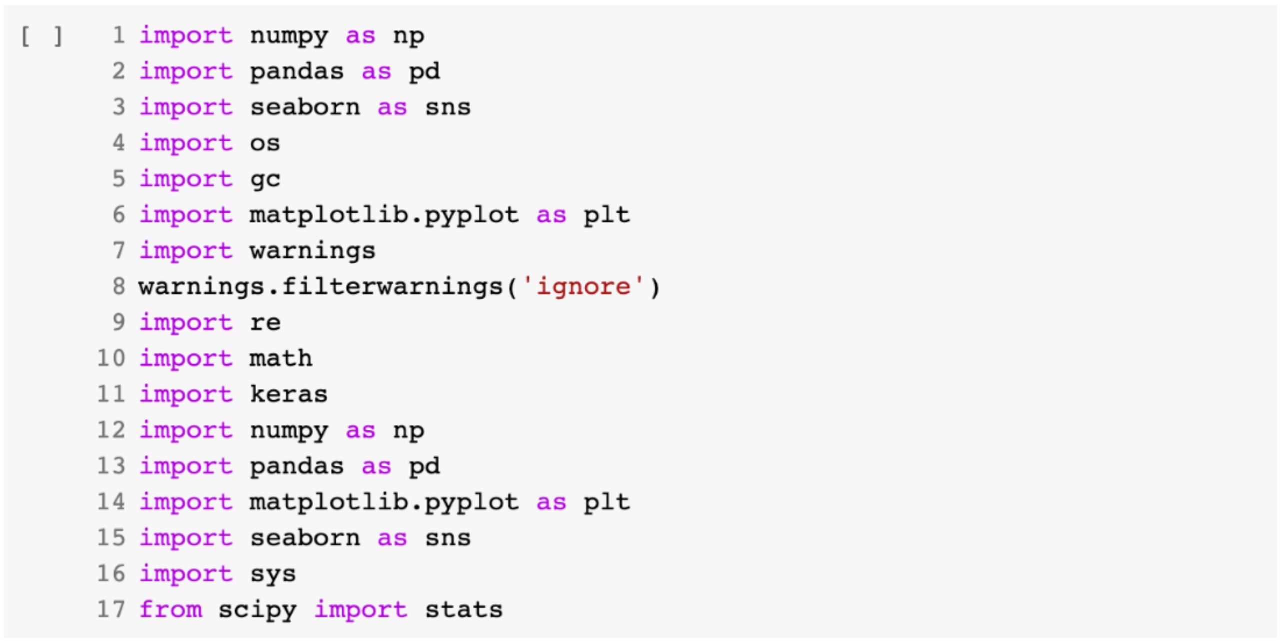
[11] W. Li, S. Li, C. Wang, C. Fang, and H. Liu,"A stacked hybrid ensemble learning model for lung cancer diagnosis",Journal of Healthcare Engineering,vol. 2021, article ID 9910532, pp. 1-9, Apr. 2021.

[12] Y. Tang, X. Li, Y. Feng, and Z. Zhang, "Ensemble deep learning for predicting lung cancer recurrence",Computer Methods and Programs in Biomedicine, vol. 195, p. 105758, Sep. 2020.

APPENDICES

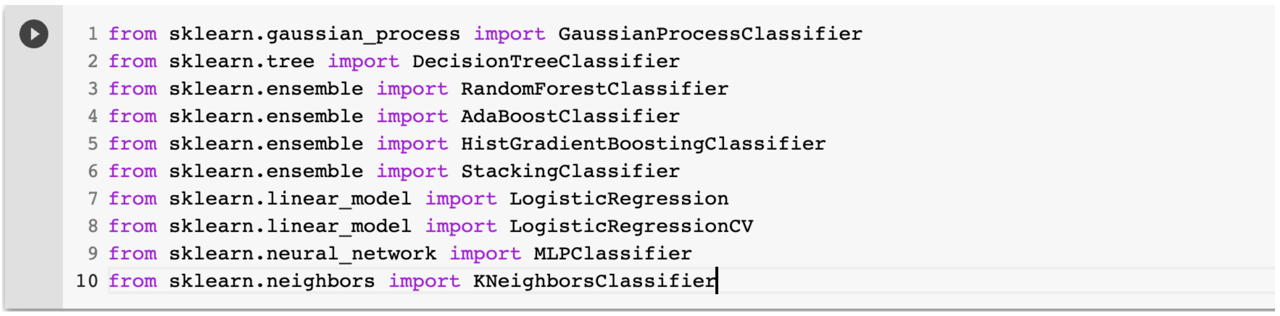
Code:

Importing Libraries:



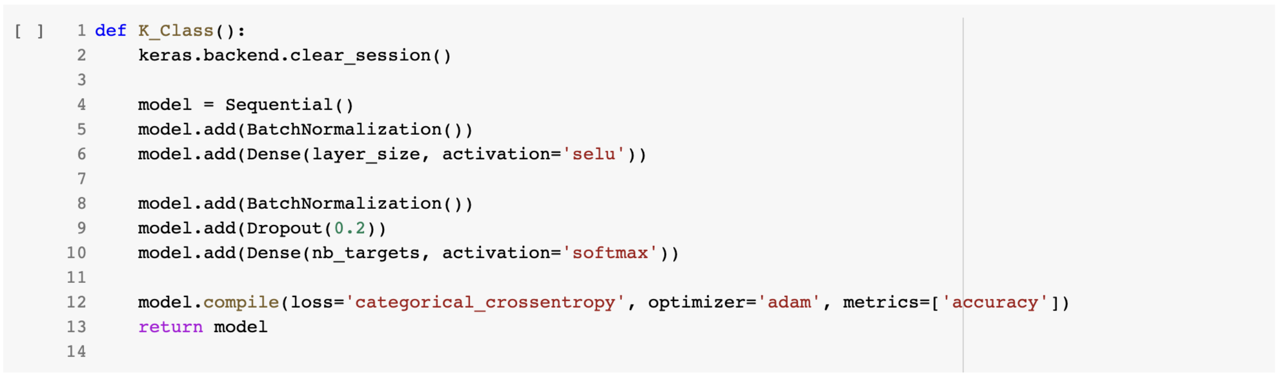
```python
import numpy as np
import pandas as pd
import seaborn as sns
import os
import gc
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import re
import math
import keras
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sys
from scipy import stats
```

Importing Models:



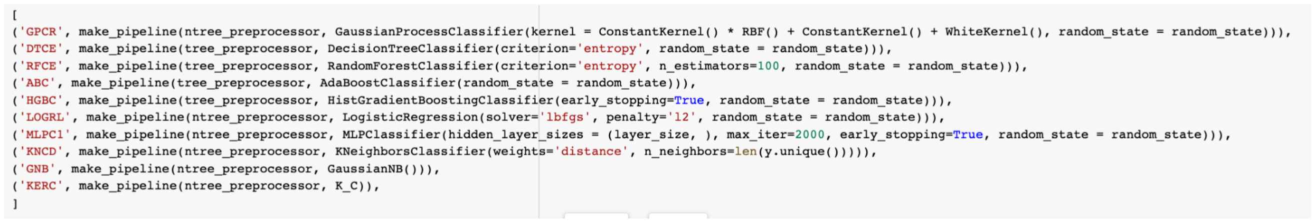
```python
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Keras Neural Network Model:



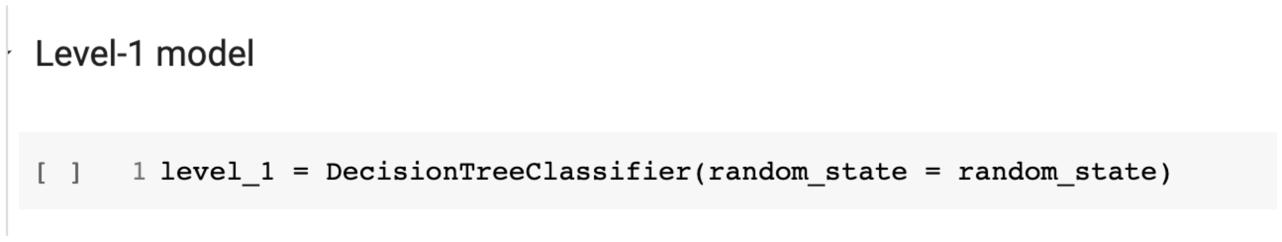
```python
def K_Class():
    keras.backend.clear_session()

    model = Sequential()
    model.add(BatchNormalization())
    model.add(Dense(layer_size, activation='selu'))

    model.add(BatchNormalization())
    model.add(Dropout(0.2))
    model.add(Dense(nb_targets, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

```

Level 0 models:



```
[
('GPCR', make_pipeline(ntree_preprocessor, GaussianProcessClassifier(kernel = ConstantKernel() * RBF() + ConstantKernel() + WhiteKernel(), random_state = random_state))),
('DTCE', make_pipeline(tree_preprocessor, DecisionTreeClassifier(criterion='entropy', random_state = random_state))),
('RFCE', make_pipeline(tree_preprocessor, RandomForestClassifier(criterion='entropy', n_estimators=100, random_state = random_state))),
('ABC', make_pipeline(tree_preprocessor, AdaBoostClassifier(random_state = random_state))),
('HGBC', make_pipeline(tree_preprocessor, HistGradientBoostingClassifier(early_stopping=True, random_state = random_state))),
('LOGRL', make_pipeline(ntree_preprocessor, LogisticRegression(solver='lbfgs', penalty='l2', random_state = random_state))),
('MLPC1', make_pipeline(ntree_preprocessor, MLPClassifier(hidden_layer_sizes = (layer_size, ), max_iter=2000, early_stopping=True, random_state = random_state))),
('KNCD', make_pipeline(ntree_preprocessor, KNeighborsClassifier(weights='distance', n_neighbors=len(y.unique()))),
('GNB', make_pipeline(ntree_preprocessor, GaussianNB())),
('KERC', make_pipeline(ntree_preprocessor, K_C)),
]
```

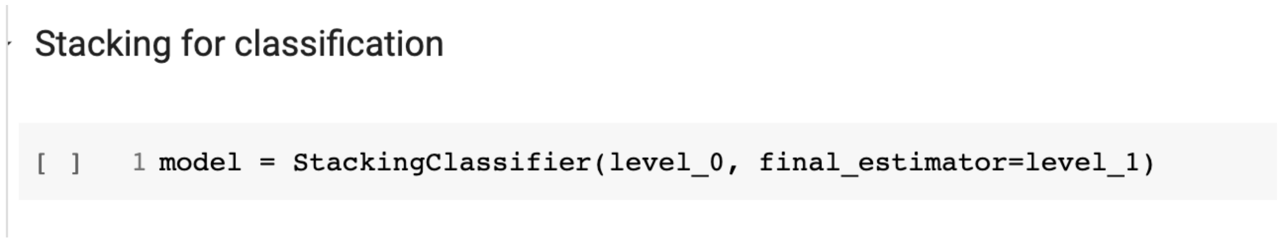
Level 1 Model:



Level-1 model

```
level_1 = DecisionTreeClassifier(random_state = random_state)
```

Stacking Model:



Stacking for classification

```
model = StackingClassifier(level_0, final_estimator=level_1)
```

Model and Feature Filtering:

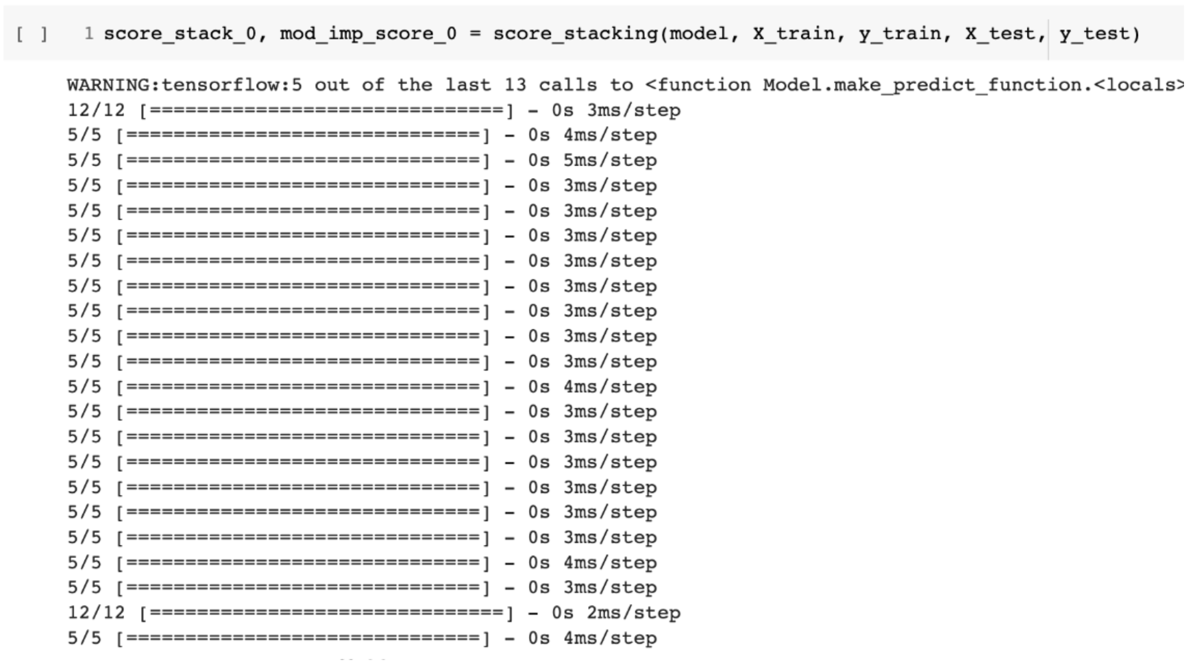
```python
def model_filtering(level_0, model_imp, nb_model, score_stack, threshold_score):


    if nb_model > len(level_0):
       nb_model = len(level_0)


    score_stack = np.delete(np.delete(score_stack, 1, axis =1), -1, axis = 0)

    score_stack = score_stack[score_stack[:,1] > threshold_score]


    if nb_model > len(score_stack):
       nb_model = len(score_stack)


    model_imp = model_imp[np.in1d(model_imp[:, 0], score_stack)]
    model_imp_f = model_imp[np.argpartition(model_imp[:,1], -nb_model)[-nb_model:]].T[0]

    return list(filter(lambda x: x[0] in model_imp_f, level_0))

def feature_filtering(feature_importance, nb_feature):

    # check nb_feature
    if nb_feature > feature_importance.shape[0]:
       nb_feature = feature_importance.shape[0]

    best_feature = feature_importance[np.argpartition(feature_importance[:,1], -nb_feature)[-nb_feature:]].T[0]
    worst_feature = list(set(feature_importance.T[0]) - set(best_feature))

    return best_feature, worst_feature
```

Model Scoring:

```python
def score_stacking_c(model, X_train, y_train, X_test, y_test):

    nb_estimators = len(model.estimators_)
    res_stack = np.empty((nb_estimators + 1, 3), dtype='object')
    m_t_x_train = model.transform(X_train)
    for j in range(nb_estimators):
        res_stack [j, 0] = [*model.named_estimators_.keys()][j]
        if m_t_x_train.shape[1] == nb_estimators:
           res_stack [j, 1] = accuracy_score(np.rint(m_t_x_train).T[j], y_train)
           res_stack [j, 2] = accuracy_score(np.rint(model.transform(X_test)).T[j], y_test)
        else:
           res_stack [j, 1] = accuracy_score(m_t_x_train.reshape((X_train.shape[0],\
                                                                  nb_estimators,\
                                                                  y_train.unique().shape[0])).argmax(axis=2).T[j],\
                                             y_train)
           res_stack [j, 2] = accuracy_score(model.transform(X_test).reshape((X_test.shape[0],\
                                                                              nb_estimators,\
                                                                              y_test.unique().shape[0])).argmax(axis=2).T[j],\
                                             y_test)
    res_stack [len(model.estimators_) , 0] = 'Stack'
    res_stack [len(model.estimators_) , 1] = accuracy_score(model.predict(X_train), y_train)
    res_stack [len(model.estimators_) , 2] = accuracy_score(model.predict(X_test), y_test)
    models = res_stack.T[0]
    score_train = res_stack.T[1]
    score_test = res_stack.T[2]
    plt.figure(figsize=(8,5))
    plt.scatter(models, score_train, label='Train')
    plt.scatter(models, score_test, label='Test')
    plt.title('Model scores: accuracy')
    plt.xticks(rotation='vertical')
    plt.legend()
    plt.show()
    return res_stack
```

```
def score_stacking_r(model, X_train, y_train, X_test, y_test):

    nb_estimators = len(model.estimators_)
    res_stack = np.empty((nb_estimators + 1, 3), dtype='object')
    m_t_x_train = model.transform(X_train)
    for j in range(nb_estimators):
        res_stack [j, 0] = [*model.named_estimators_.keys()][j]
        res_stack [j, 1] = r2_score(np.rint(m_t_x_train).T[j], y_train)
        res_stack [j, 2] = r2_score(np.rint(model.transform(X_test)).T[j], y_test)
    res_stack [len(model.estimators_) , 0] = 'Stack'
    res_stack [len(model.estimators_) , 1] = r2_score(model.predict(X_train), y_train)
    res_stack [len(model.estimators_) , 2] = r2_score(model.predict(X_test), y_test)
    models = res_stack.T[0]
    score_train = res_stack.T[1]
    score_test = res_stack.T[2]
    plt.figure(figsize=(8,5))
    plt.scatter(models, score_train, label='Train')
    plt.scatter(models, score_test, label='Test')
    plt.title('Model scores: r2')
    plt.xticks(rotation='vertical')
    plt.legend()
    plt.show()
    return res_stack
```

```
def score_stacking(model, X_train, y_train, X_test, y_test):

    if is_classifier(model):
       res_stack = score_stacking_c(model, X_train, y_train, X_test, y_test)
    else:
       res_stack = score_stacking_r(model, X_train, y_train, X_test, y_test)
    nb_estimators = len(model.estimators_)
    res_level_0 = res_stack[0:nb_estimators]
    mod_imp = np.delete(res_level_0[res_level_0[:, 2].argsort()], 1, axis=1)
    mod_imp.T[1] = mod_imp.T[1] / np.sum(mod_imp.T[1])
    fig, ax = plt.subplots()
    ax.barh(mod_imp.T[0], mod_imp.T[1])
    ax.set_title("Model Importance according to performance")
    fig.tight_layout()
    plt.show()
    return res_stack, mod_imp
```



```
[ ]   score_stack_0, mod_imp_score_0 = score_stacking(model, X_train, y_train, X_test, y_test)
```

```
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_predict_function.<locals>
12/12 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 4ms/step
5/5 [==============================] - 0s 5ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 4ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 3ms/step
5/5 [==============================] - 0s 4ms/step
5/5 [==============================] - 0s 3ms/step
12/12 [==============================] - 0s 2ms/step
5/5 [==============================] - 0s 4ms/step
```

Model Importance:

```
def model_importance_c(model, level_1_model):

    level_0 = np.array(list(model.named_estimators_.keys()))
    n_classes = model.classes_.shape[0]
    n_models = len(model.estimators_)
    model_coeff = find_coeff(model.final_estimator_)

    if level_1_model == 'tree':
       if len(model_coeff) == n_models:
          coeff = model_coeff.reshape(n_models)
       else:
          coeff = sum(model_coeff.reshape(n_classes,n_models))

    if level_1_model == 'regression':
       if len(model_coeff[0]) == n_models:
          coeff = model_coeff.reshape(n_models)
       else:
          coeff = sum(model_coeff.reshape(n_classes,n_models,n_classes)[i].T[i] for i in range(n_classes))

    model_importance = np.empty((len(level_0), 2), dtype='object')
    for ind in range(len(level_0)):
        model_importance[ind, 0] = level_0[ind]
        model_importance[ind, 1] = np.abs(coeff[ind])
    return model_importance[model_importance[:, 1].argsort()]
```

```
def model_importance_r(model, level_1_model):

    level_0 = np.array(list(model.named_estimators_.keys()))
    coeff = find_coeff(model.final_estimator_)
    model_importance = np.empty((len(level_0), 2), dtype='object')
    for ind in range(len(level_0)):
        model_importance[ind, 0] = level_0[ind]
        model_importance[ind, 1] = np.abs(coeff[ind])
    return model_importance[model_importance[:, 1].argsort()]

def plot_model_importance(model, level_1_model):

    if is_classifier(model):
       mod_imp = model_importance_c(model, level_1_model)
    else:
       mod_imp = model_importance_r(model, level_1_model)
    mod_imp.T[1] = mod_imp.T[1] / np.sum(mod_imp.T[1])
    fig, ax = plt.subplots()
    ax.barh(mod_imp.T[0], mod_imp.T[1])
    ax.set_title("Model Importance according to aggragator coefficients")
    fig.tight_layout()
    plt.show()
    return mod_imp
```