American International University -Bangladesh

Faculty of Science and Technology (FST)

Machine Learning Project Report

Course Name: Machine Learning	
Course Code: 01668	**Section:** A
Instructor Name: DR. MD. ASRAF ALI	
Semester: Spring 2023-24	

Submitted by:

Group No: 2	
STUDENT NAME	**STUDENT ID**
K M Yeaser Arafat	21-44933-2
Gobinda Goswamee	21-44984-2
MD. Abu Raihan	21-45789-3
Devdoot Parial	21-45061-2

Submit Date:	
Project Title:	Lung cancer analysis, prediction, feature selection & detection using machine learning.

Outline

1. Introduction:

1.1 Background

Machine learning methods have revolutionized lung cancer research by analyzing complex datasets, including medical images and genetic profiles, to enhance early detection, accurate classification, and personalized treatment strategies.

1.2 Problem Statement (analyzed by existing solution)

Certainly, here's a potential problem statement analyzed through existing solutions for a thesis report on lung cancer analysis, prediction, classification, and detection using machine learning:

Problem Statement:
Lung cancer remains one of the most prevalent and deadly forms of cancer worldwide, with late-stage diagnosis significantly reducing patient survival rates. Despite advancements in medical imaging and molecular diagnostics, challenges persist in accurately detecting, classifying, and predicting the progression of lung cancer. Traditional diagnostic methods often rely on subjective interpretation and may lack the sensitivity required for early detection, leading to missed opportunities for timely intervention.

Analyzing Existing Solutions:
Existing solutions in lung cancer analysis have increasingly leveraged machine learning techniques to address diagnostic and prognostic challenges. These solutions encompass various approaches, including:

i. **Medical Imaging Analysis:** Machine learning algorithms applied to chest X-rays, CT scans, and MRI images have shown promise in detecting subtle abnormalities indicative of lung cancer, enabling earlier diagnosis and intervention. Several studies have demonstrated the efficacy of convolutional neural networks (CNNs) and other deep learning architectures in automating the interpretation of medical images and improving diagnostic accuracy.

ii. **Genomic and Molecular Profiling:** Molecular profiling of lung cancer tumors has facilitated the identification of biomarkers associated with disease progression and treatment response. Machine learning models trained on genomic, transcriptomic, and proteomic data have been developed to classify lung cancer subtypes, predict treatment outcomes, and guide personalized therapeutic strategies. Integration of multi-omics data using advanced machine learning techniques has further enhanced the predictive power of these models.

iii. **Clinical Data Integration:** Machine learning approaches have been employed to integrate diverse clinical data sources, including electronic health records (EHRs),

pathology reports, and patient demographics, to improve lung cancer diagnosis and prognosis. By analyzing large-scale datasets, these models can identify patterns and correlations that inform risk stratification, treatment selection, and prognostic assessment.

iv. **Interpretability and Validation:** Efforts have been made to enhance the interpretability and validation of machine learning models in lung cancer analysis. Explainable AI techniques have been developed to elucidate the underlying decision-making processes of black-box models, fostering trust and acceptance among healthcare providers. Additionally, rigorous validation studies have been conducted to assess the generalizability and clinical relevance of machine learning-based diagnostic and prognostic tools.

However, despite these advancements, several challenges persist, including:

- **Data Quality and Bias:** Limited availability of high-quality annotated datasets and potential biases in training data pose challenges to the development and validation of robust machine learning models for lung cancer analysis.

- **Clinical Integration:** Integrating machine learning solutions into routine clinical workflows and ensuring seamless interoperability with existing healthcare systems remain logistical challenges that require multidisciplinary collaboration and stakeholder engagement.

- **Ethical Considerations:** Ethical concerns regarding patient privacy, informed consent, and algorithmic transparency must be carefully addressed to ensure the responsible and ethical deployment of machine learning technologies in lung cancer diagnosis and treatment.

Therefore, this thesis aims to address these challenges by proposing novel machine learning approaches for lung cancer analysis, with a focus on improving diagnostic accuracy, prognostic prediction, and clinical utility.
This problem statement provides a comprehensive overview of the current landscape of lung cancer analysis using machine learning, highlighting both the advancements made and the persistent challenges that need to be addressed. It sets the stage for the proposed research objectives and methodology to tackle these challenges effectively.

1.3 Objective

Certainly, here are some objectives for lung cancer analysis, prediction, classification, and detection using machine learning:

i. **Early Detection:** Develop machine learning models capable of accurately detecting early signs of lung cancer in medical imaging data, such as X-rays, CT scans, and MRIs, to enable timely intervention and improve patient outcomes.

ii. **Risk Stratification:** Create predictive models that assess an individual's risk of developing lung cancer based on factors such as smoking history, genetic predisposition, environmental exposure, and demographic characteristics, to facilitate targeted screening and preventive interventions.

iii. **Subtype Classification:** Build machine learning algorithms capable of accurately classifying lung cancer subtypes, such as non-small cell lung cancer (NSCLC) and small cell lung cancer (SCLC), using molecular and genetic data, to guide treatment selection and improve patient prognosis.

iv. **Treatment Response Prediction:** Develop models that predict patient response to different treatment modalities, including chemotherapy, radiation therapy, immunotherapy, and targeted therapy, based on clinical, molecular, and imaging data, to personalize treatment plans and optimize therapeutic outcomes.

v. **Prognostic Modeling:** Create prognostic models that predict the progression of lung cancer and estimate patient survival probabilities using clinical, pathological, and molecular features, to assist clinicians in treatment planning and patient counseling.

vi. **Integration of Multi-Omics Data:** Incorporate diverse omics data, including genomics, transcriptomics, proteomics, and metabolomics, into machine learning models to improve the accuracy of lung cancer diagnosis, prognosis, and treatment prediction.

vii. **Interpretability and Explainability:** Ensure that machine learning models used for lung cancer analysis are interpretable and explainable, enabling clinicians to understand the rationale behind predictions and fostering trust in the decision-making process.

viii. **Validation and Clinical Translation:** Validate machine learning models using independent datasets and rigorous evaluation metrics to assess their generalizability and clinical utility, facilitating their integration into routine clinical practice for the benefit of patients.

ix. **Ethical Considerations:** Address ethical concerns related to patient privacy, informed consent, data security, and potential biases in machine learning algorithms, ensuring that lung cancer analysis using machine learning is conducted ethically and responsibly.

x. **Collaboration and Knowledge Sharing:** Foster interdisciplinary collaboration between clinicians, data scientists, bioinformaticians, and ethicists to leverage expertise from diverse domains and promote knowledge sharing in the development and deployment of machine learning solutions for lung cancer analysis.

Despite these promising applications, it's essential to address challenges such as data bias, interpretability, and ethical considerations to ensure the responsible and effective use of machine learning in lung cancer analysis. Collaborations between clinicians, data scientists, and ethicists are crucial for developing and deploying machine learning models that benefit patients while upholding ethical standards and patient privacy.

1.4 Methods

Lung Cancer Analysis: Correlation Heatmap, preprocessing for classification: Logistic Regression Model, Decision Tree, Naive Bayes, Neural network architecture classification (simple Multi-Layer Perceptron (MLP) architecture)

Lung Cancer Prediction: logistic regression, decision tree, k-nearest neighbor, Gaussian naive Bayes, multinomial naive Bayes, support vector classifier, random forest

Lung Cancer Detection: Random Forest, NNA (Neural Network Architecture).

2. Methodology

2.1 Data Collection Procedure

Certainly, here's a concise procedure for collecting a lung cancer dataset from Kaggle as a CSV file for a machine learning thesis:

i. **Search Kaggle:** Use keywords like "lung cancer" to find relevant datasets.
ii. **Evaluate Quality:** Check completeness, accuracy, and relevance.
iii. **Download CSV:** Select the dataset and download it in CSV format.
iv. **Review Licensing:** Ensure compliance with licensing terms.
v. **Verify Data:** Check data integrity post-download.
vi. **Document Source:** Record dataset details for referencing.
vii. **Preprocess Data:** Optionally clean, handle missing values, and transform variables.

This succinct procedure covers the essential steps for acquiring a lung cancer dataset from Kaggle for machine learning research.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
GENDER	AGE	SMOKING	YELLOW_F	ANXIETY	PEER_PRE	CHRONIC	FATIGUE	ALLERGY	WHEEZINC	ALCOHOL	COUGHIN	SHORTNE	SWALLOW	CHEST PAI	LUNG_CANCER	
M	69	1	2	2	1	1	2	1	2	2	2	2	2	2	YES	
M	74	2	1	1	1	2	2	2	1	1	1	2	2	2	YES	
F	59	1	1	1	2	1	2	1	2	1	2	2	1	2	NO	
M	63	2	2	2	1	1	1	1	1	2	1	1	2	2	NO	
F	63	1	2	1	1	1	1	1	2	1	2	2	1	1	NO	
F	75	1	2	1	1	2	2	2	2	1	2	2	1	1	YES	
M	52	2	1	1	1	1	2	1	2	2	2	2	1	2	YES	
F	51	2	2	2	2	1	2	2	1	1	1	2	2	1	YES	
F	68	2	1	2	1	1	2	1	1	1	1	1	1	1	NO	
M	53	2	2	2	2	2	1	2	1	2	1	1	2	2	YES	
F	61	2	2	2	2	2	2	1	2	1	2	2	2	1	YES	
M	72	1	1	1	1	2	2	2	2	2	2	2	1	2	YES	
F	60	2	1	1	1	1	2	1	1	1	1	2	1	1	NO	
M	58	2	1	1	1	1	2	2	2	2	2	2	1	2	YES	
M	69	2	1	1	1	1	1	2	2	2	2	1	1	2	NO	
F	48	1	2	2	2	2	2	2	2	1	2	2	2	1	YES	
M	75	2	1	1	1	2	1	2	2	2	2	2	1	2	YES	
M	57	2	2	2	2	2	1	1	1	2	1	1	2	2	YES	
F	68	2	2	2	2	2	2	1	1	1	2	2	1	1	YES	
F	61	1	1	1	1	2	2	1	1	1	1	2	1	1	NO	
F	44	2	2	2	2	2	2	1	1	1	1	2	2	1	YES	
F	64	1	2	2	2	1	1	2	2	1	2	1	2	1	YES	
F	21	2	1	1	1	2	2	2	1	1	1	2	1	1	NO	
M	60	2	1	1	1	1	2	2	2	2	2	2	1	2	YES	
M	72	2	2	2	2	2	1	2	2	2	2	1	2	2	YES	
M	65	1	2	2	1	1	2	1	2	2	2	2	2	2	YES	

Table 1. Represent the csv dataset which is collected from Kaggle.

Feature	Description
GENDER	M[Male], F[Female]
AGE	Age of patients
SMOKING	2 [Yes] , 1 [No]
YELLOW_FINGURE	2 [Yes] , 1 [No]
ANXIETY	2 [Yes] , 1 [No]
PEER_PRESSURE	2 [Yes] , 1 [No]
CHRONIC DISEASE	2 [Yes] , 1 [No]
FATIGUE	2 [Yes] , 1 [No]
ALLERGY	2 [Yes] , 1 [No]
WHEEZING	2 [Yes] , 1 [No]
ALCOHOL COMSUMING	2 [Yes] , 1 [No]
COUGHING	2 [Yes] , 1 [No]
SHORTNESS OF BREATH	2 [Yes] , 1 [No]
SWALLOWING DIFFICULTY	2 [Yes] , 1 [No]
CHEST PAIN	2 [Yes] , 1 [No]
LUNG CANCER	YES [Positive] , NO [Negative]

Table 2. Description of attributes for the dataset collected from Kaggle [5]

2.2. Data Validation Procedure

Data validation is a critical procedure in ensuring the accuracy, completeness, and consistency of data. Here's a generalized procedure for data validation:

1. **Define Requirements:** Clearly outline the requirements and standards for the data. This includes specifying acceptable formats, ranges, and any business rules that apply.

2. Data Collection: Gather the data from various sources, ensuring it aligns with the defined requirements.

3. Data Cleaning: Before validation, clean the data to remove any inconsistencies, errors, or duplicates. This step might involve parsing, standardizing formats, and resolving missing or erroneous values.

4. Validation Rules Creation: Develop validation rules based on the defined requirements. These rules can include checks for data type, format, range, uniqueness, and referential integrity.

5. Manual Inspection: Some aspects of data validation may require human inspection. Review the data manually to identify any obvious errors or inconsistencies that automated processes might miss.

6. Automated Validation: Utilize software tools or scripts to automate validation processes. These tools can efficiently apply validation rules across large datasets, flagging any data that fails to meet the specified criteria.

7. Error Reporting: Develop a system for reporting validation errors. Clearly communicate any issues found during the validation process, including details such as the nature of the error, its location in the data, and potential solutions.

8. Error Resolution: Once errors are identified, take appropriate action to resolve them. This might involve correcting erroneous data, reconciling inconsistencies, or revising validation rules as necessary.

9. Documentation: Document the validation process thoroughly, including the validation rules, any manual interventions, and the outcomes of the validation process. This documentation serves as a reference for future validation efforts and helps ensure consistency over time.

10. Regular Review: Establish a schedule for regular review and validation of data. Data quality can degrade over time, so periodic validation helps to maintain data integrity and reliability.

12. Continuous Improvement: Continuously evaluate and refine the data validation procedures to adapt to changing requirements, technology, and business needs. Aim for a cycle of continuous improvement to enhance data quality over time.

2.3 Data Prepossessing Technique

Data preprocessing is a crucial step in data analysis and machine learning. It involves transforming raw data into a format that is more suitable for downstream analysis. Here are some common data preprocessing techniques:

1. Data Cleaning:

- Handling Missing Values: Methods include imputation (replacing missing values with a calculated estimate), deletion (removing records or features with missing values) or using algorithms that can handle missing data.
- Outlier Detection and Treatment: Identifying and handling outliers through techniques such as statistical methods, clustering, or domain knowledge.

2. Data Transformation:

- Scaling/Normalization: Scaling features to a similar range to prevent features with larger scales from dominating algorithms that rely on distance metrics. Common methods include Min-Max scaling and Z-score standardization.
- Log Transformation: Transforming skewed data distributions to more closely resemble a normal distribution, which can improve the performance of certain algorithms.
- Encoding Categorical Variables: Converting categorical variables into numerical representations suitable for machine learning algorithms. Techniques include one-hot encoding, label encoding, and target encoding.
- Feature Engineering: Creating new features or transforming existing ones to capture more meaningful information or improve model performance.

3. Data Reduction:

- Dimensionality Reduction: Techniques such as Principal Component Analysis (PCA) or t-distributed Stochastic Neighbor Embedding (t-SNE) to reduce the number of features while preserving the most important information.
- Feature Selection: Selecting the most relevant features based on criteria such as feature importance, correlation, or domain knowledge.

4. Data Integration:

- Combining data from multiple sources into a unified dataset, resolving inconsistencies in naming conventions, formats, or units of measurement.

5. Data Discretization:

- Binning: Grouping continuous numerical data into discrete bins or intervals.
- Discretization: Converting continuous features into categorical features to simplify the data and potentially improve model interpretability.

6. Data Augmentation (commonly used in image and text data):

- Generating synthetic data points to increase the size of the dataset, improve model generalization, and address class imbalances.

7. Text Preprocessing (specific to text data):

- Tokenization: Splitting text into individual words or tokens.
- Stopword Removal: Removing common words (e.g., "the", "is", "and") that do not carry significant meaning.
- Lemmatization/Stemming: Reducing words to their root form to normalize variations (e.g., "running" → "run").
- Vectorization: Converting text data into numerical representations, such as bag-of-words or TF-IDF vectors, suitable for machine learning algorithms.

Each of these techniques plays a vital role in preparing data for analysis or modeling, and the choice of techniques depends on the characteristics of the dataset and the specific goals of the analysis or modeling task.

2.4 Feature Extraction Technique

Feature extraction is a crucial step in data analysis, particularly when dealing with complex datasets. One effective method for extracting features is through the use of heatmaps. Heatmaps visually represent data using colors to indicate the magnitude of values within a matrix or grid. In the context of feature extraction, heatmaps can highlight patterns, correlations, or important relationships within the data.

By plotting features against each other and color-coding the resulting matrix based on their relationships (such as correlation coefficients), heatmaps provide a concise and intuitive way to identify significant features. High correlations or patterns can stand out as clusters of similarly colored cells, indicating strong associations between specific features. Conversely, low correlations or lack of patterns can also be easily identified, helping to pinpoint less relevant or redundant features.

Heatmaps streamline the feature selection process by offering a clear visual representation of the data's underlying structure. This allows analysts and data scientists to make informed decisions about which features to include in further analysis or modeling efforts. Ultimately, by leveraging the power of heatmaps for feature extraction, organizations can uncover valuable insights and drive more effective decision-making processes.

2.5 Classification Algorithms

Here's a brief description of each classification algorithm:

1. Logistic Regression Model:

- Logistic regression is a linear classification model used to predict the probability of a binary outcome based on one or more predictor variables.
- It models the relationship between the independent variables and the probability of a particular outcome using the logistic function, which maps the output to the range [0, 1].
- Despite its name, logistic regression is primarily used for classification tasks rather than regression tasks.

2. Decision Tree:

- A decision tree is a tree-like structure used for classification and regression tasks.
- It recursively splits the data into subsets based on the value of features, with each internal node representing a feature, each branch representing a decision based on that feature, and each leaf node representing a class label or regression value.
- Decision trees are interpretable and easy to visualize, making them useful for understanding the decision-making process.

3. Naive Bayes:

- Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem and the assumption of conditional independence between features.
- It calculates the probability of each class given the observed features and selects the class with the highest probability as the predicted class.
- Despite its simplicity and the naive assumption of feature independence, Naive Bayes often performs well, especially on text classification tasks.

4. Neural Network (Simple Multi-Layer Perceptron):

- A neural network is a versatile machine learning model inspired by the structure and function of the human brain.
- A simple Multi-Layer Perceptron (MLP) architecture consists of an input layer, one or more hidden layers, and an output layer.
- Each neuron in the network applies an activation function to a weighted sum of its inputs, transforming the input data through nonlinear transformations to learn complex patterns.
- MLPs are capable of learning complex relationships in the data and are widely used for classification tasks, especially when dealing with high-dimensional data or non-linear decision boundaries.

These classification algorithms offer different trade-offs in terms of interpretability, computational complexity, and performance on different types of datasets. The choice of algorithm depends on factors such as the nature of the data, the interpretability requirements, and the desired balance between accuracy and computational efficiency.

2.6 Data Analysis Techniques

Data analysis techniques encompass a variety of methods used to explore, interpret, and draw insights from data. Here's a more detailed description of some common techniques:

1. Descriptive Statistics:

- Descriptive statistics provide summaries of the main features of a dataset. These include measures of central tendency (mean, median, mode), measures of dispersion (variance, standard deviation, range), and measures of distribution (skewness, kurtosis).

2. Exploratory Data Analysis (EDA):
- EDA involves visualizing and exploring data to understand its structure, patterns, and relationships. Techniques include histograms, box plots, scatter plots, and correlation matrices to uncover trends, outliers, and associations between variables.

3. Inferential Statistics:
- Inferential statistics are used to make inferences and predictions about a population based on a sample of data. This includes hypothesis testing, confidence intervals, and regression analysis to assess relationships and draw conclusions about the broader population.

4. Predictive Modeling:
- Predictive modeling uses statistical and machine learning algorithms to make predictions based on historical data. Techniques range from simpler models like linear regression and logistic regression to more complex algorithms such as decision trees, random forests, support vector machines, and neural networks.

5. Time Series Analysis:
- Time series analysis focuses on analyzing time-ordered data points to identify patterns, trends, and seasonal effects. Techniques include decomposition, autocorrelation analysis, and forecasting methods like ARIMA and exponential smoothing.

6. Cluster Analysis:
- Cluster analysis groups similar data points together based on their characteristics or features. Common techniques include k-means clustering, hierarchical clustering, and density-based clustering to identify natural groupings within the data.

7. Dimensionality Reduction:
- Dimensionality reduction techniques aim to reduce the number of variables in a dataset while preserving important information. Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are popular methods for reducing the dimensionality of data.

8. Text Mining and Natural Language Processing (NLP):
- Text mining and NLP techniques analyze and extract insights from unstructured text data. This includes tokenization, sentiment analysis, topic modeling (e.g., Latent Dirichlet Allocation), and named entity recognition to uncover patterns and sentiments within text data.

9. Data Visualization:
- Data visualization techniques present data visually through charts, graphs, and interactive dashboards to aid in understanding and communication. Effective visualization enhances data exploration, interpretation, and storytelling.

By employing these data analysis techniques, analysts and data scientists can gain valuable insights, make data-driven decisions, and uncover meaningful patterns and trends within their data. Each technique serves a specific purpose and can be applied iteratively or in combination to extract actionable insights from diverse datasets.

$$accuracy = \frac{number\ of\ correct\ \ predictions}{(total\ number\ of\ predictions)}$$

$$F1\ score = 2\ \times \frac{(Precision\ \times recall)}{(precision + recall)}$$

2.7 Block Diagram of Proposed Model

Here's a description of a proposed model using a neural network architecture (NNA) depicted in a block diagram:

The proposed model is a neural network architecture designed for image classification tasks. It consists of several layers:

1. Input Layer: This layer represents the input data, which consists of images. Each pixel in the image corresponds to a node in the input layer.

2. Hidden Layers: The model includes one or more hidden layers, each comprising multiple neurons. These hidden layers apply non-linear transformations to the input data, allowing the network to learn complex patterns and features from the images.

3. Output Layer: The output layer produces the final classification results. Each neuron in this layer corresponds to a specific class label, and the output values represent the model's confidence scores for each class.

4. Activation Functions: Between each layer, activation functions are applied to introduce non-linearity into the model. Common activation functions include ReLU (Rectified Linear Unit) for hidden layers and SoftMax for the output layer in classification tasks.

5. Weights and Biases: Connections between neurons in adjacent layers are represented by weights, which are learned during the training process. Biases are added to each neuron to shift the activation function's threshold.

6. Loss Function: The loss function measures the difference between the predicted outputs and the true labels. It quantifies the model's performance during training and guides the optimization process.

7. Optimizer: The optimizer adjusts the weights and biases of the neural network based on the gradients of the loss function to minimize prediction error. Common optimizers include stochastic gradient descent (SGD), Adam, and RMSprop.

8. Training Process: The model is trained iteratively using a labeled dataset. During training, input images are forward propagated through the network to generate predictions, and then backpropagated to update the model's parameters based on the calculated gradients.

This block diagram illustrates the flow of information through the neural network architecture, showing how the input data is transformed through successive layers to produce the final classification results. The model's performance is evaluated based on its ability to accurately classify images into their respective categories.

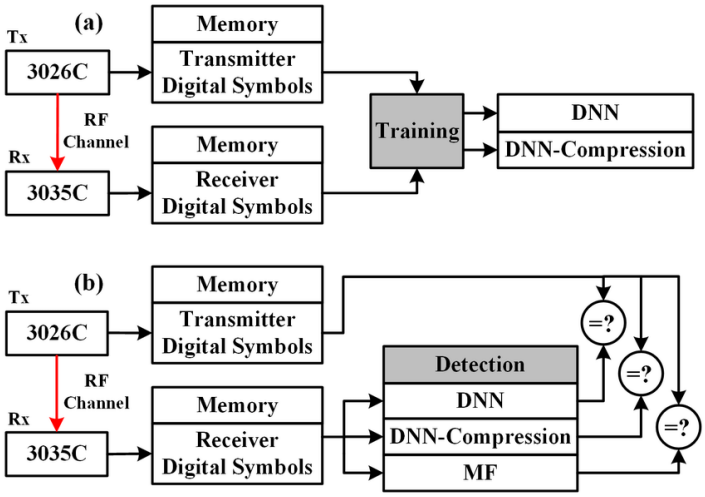


Fig. 1 Block Diagram of proposed methodology

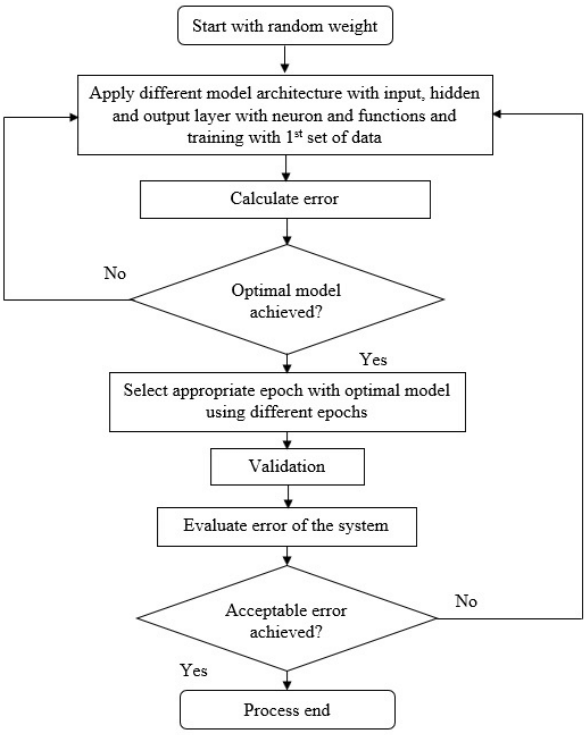


Fig 3. Workflow diagram of proposed methodology

2.8 Experimental Setup and Implementations

Experimental setup and implementation refer to the process of designing, conducting, and executing experiments to test hypotheses, evaluate models, or validate algorithms. Here's a brief description:

1. Experimental Setup:

- Define the research question or hypothesis to be tested.
- Identify the variables, parameters, and factors involved in the experiment.
- Design the experimental protocol, including the experimental conditions, treatments, and control groups.
- Specify the data collection methods, instrumentation, and measurement techniques.
- Determine the sample size, experimental duration, and other logistical considerations.

2. Implementations:

- Implement the experimental setup according to the designed protocol.
- Collect data using appropriate tools, equipment, or software.
- Conduct the experiment under controlled conditions to minimize confounding factors and biases.
- Monitor and record data systematically, ensuring accuracy and reliability.
- Analyze the collected data using statistical methods, machine learning algorithms, or other analytical techniques.
- Interpret the results and draw conclusions based on the experimental findings.
- Communicate the experimental methodology, results, and conclusions through reports, presentations, or publications.

Experimental setup and implementation are essential steps in scientific research, allowing researchers to systematically investigate phenomena, test hypotheses, and advance knowledge in their respective fields. Proper planning, execution, and analysis are critical to ensuring the validity and reproducibility of experimental findings.

3. Results and Discussion

The consequences of the machine learning classifiers have been analyzed by evaluating the performance of the algorithms, as mentioned below.

3.1 Results Analysis by comparison existing solution

Analyze the results based on the provided classification reports for each model:

1. Logistic Regression Model:

- Precision for class 0: 1.00, Precision for class 1: 0.92
- Recall for class 0: 0.67, Recall for class 1: 1.00
- F1-score for class 0: 0.80, F1-score for class 1: 0.96
- Accuracy: 92.86%
- Strengths: Achieved high precision, recall, and F1-score for both classes. Overall high accuracy.
- Weaknesses: Slightly lower recall for class 0 compared to class 1.

2. Decision Tree:

- Precision for class 0: 0.83, Precision for class 1: 0.86
- Recall for class 0: 0.42, Recall for class 1: 0.98
- F1-score for class 0: 0.56, F1-score for class 1: 0.91
- Accuracy: 85.71%
- Strengths: Achieved high precision, recall, and F1-score for class 1. Overall decent accuracy.
- Weaknesses: Lower recall for class 0 compared to class 1. Lower precision and F1-score for class 0.

3. Naive Bayes:

- Precision for class 0: 0.89, Precision for class 1: 0.91
- Recall for class 0: 0.67, Recall for class 1: 0.98
- F1-score for class 0: 0.76, F1-score for class 1: 0.95
- Accuracy: 91.07%
- Strengths: High precision, recall, and F1-score for both classes. Overall high accuracy.
- Weaknesses: Slightly lower recall for class 0 compared to class 1.

4. Neural Network Classifier (MLP):

- Precision for class 0: 1.00, Precision for class 1: 0.88
- Recall for class 0: 0.50, Recall for class 1: 1.00
- F1-score for class 0: 0.67, F1-score for class 1: 0.94
- Accuracy: 89.29%
- Strengths: High precision for class 0 and high recall for class 1. Overall decent accuracy.
- Weaknesses: Lower recall for class 0 compared to class 1.

In summary, each model has its strengths and weaknesses based on precision, recall, and F1-score for each class. Naive Bayes performed consistently well across all metrics, achieving the highest overall accuracy. Logistic regression also performed well with high precision and recall for both classes. Decision tree and neural network classifier showed some variability in performance, with strengths in certain metrics but weaknesses in others. The choice of model

depends on specific requirements and priorities such as interpretability, computational efficiency, and the importance of correctly identifying positive and negative cases.

3.2 Results validation by Graphical Representation

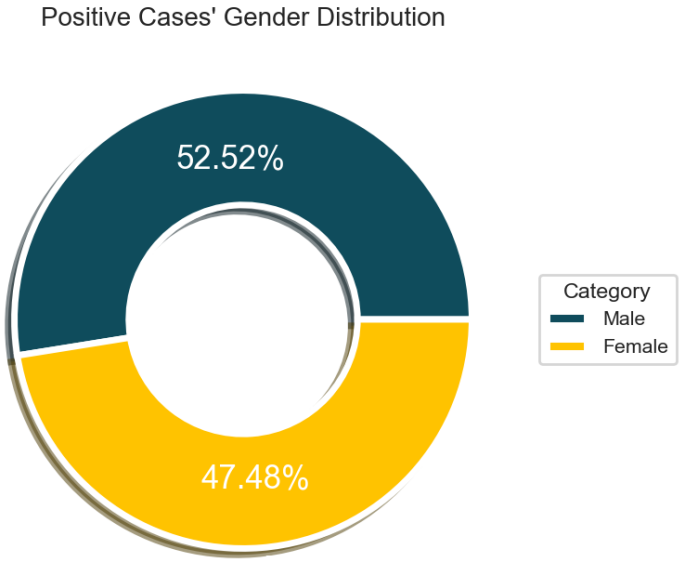
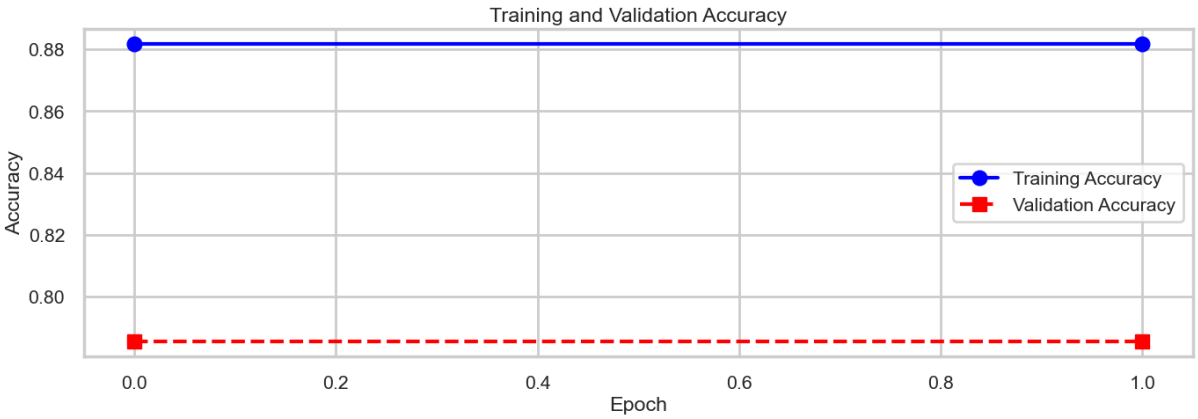
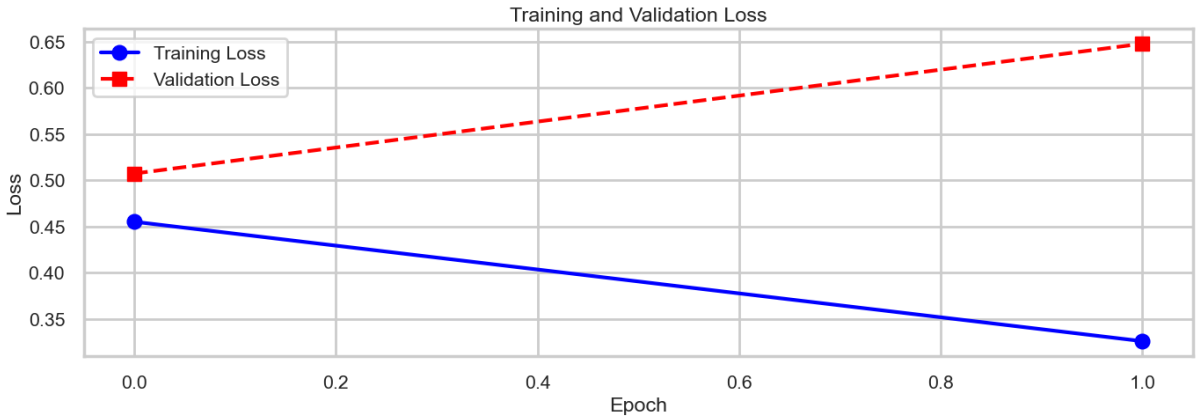
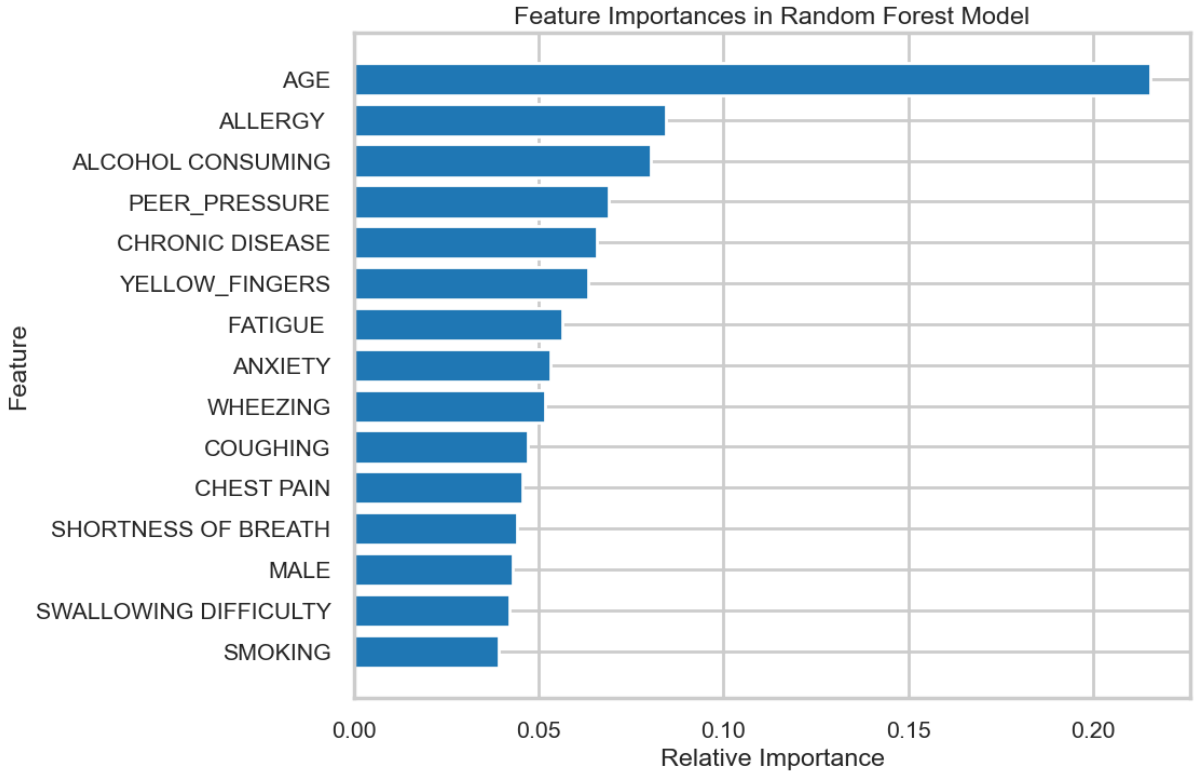


Fig 3.2.1 Gender Distribution of positive cases


Effect of Smoking
SMOKING
Yes
No
Quantity
100
80
60
40
20
0
71
54
60
53
Male
Female
Gender
Effect of Alcohol Consuming
ALCOHOL CONSUMING
Yes
No
101
24
44
69
Male
Female
Gender
Yellow_Fingers
Yes
No
Total
80
60
40
20
0
62
63
84
29
Male
Female
GENDER
Anxiety
Yes
No
52
73
73
40
Male
Female
GENDER
Chronic Disease
Yes
No
57
68
74
39
Male
Female
GENDER
Chest Pain
Yes
No
94
31
48
65
Male
Female
GENDER
Fatigue
Yes
No
84
41
81
32
Male
Female
GENDER
Wheezing
Yes
No
Total
80
60
40
20
0
81
44
61
52
Male
Female
GENDER
Coughing
Yes
No
85
40
64
49
Male
Female
GENDER
Shortness Of Breath
Yes
No
80
45
73
40
Male
Female
GENDER
Swallowing Difficulty
Yes
No
59
66
65
48
Male
Female
GENDER
Allergy
Yes
No
85
40
61
52
Male
Female
GENDER


Training and Validation Accuracy
Accuracy
0.88
0.86
0.84
0.82
0.80
0.0
0.2
0.4
0.6
0.8
1.0
Epoch
Training Accuracy
Validation Accuracy


Training and Validation Loss
Loss
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.0
0.2
0.4
0.6
0.8
1.0
Epoch
Training Loss
Validation Loss


Feature Importances in Random Forest Model
Feature
AGE
ALLERGY
ALCOHOL CONSUMING
PEER_PRESSURE
CHRONIC DISEASE
YELLOW_FINGERS
FATIGUE
ANXIETY
WHEEZING
COUGHING
CHEST PAIN
SHORTNESS OF BREATH
MALE
SWALLOWING DIFFICULTY
SMOKING
0.00
0.05
0.10
0.15
0.20
Relative Importance

4. Conclusion and Future Recommendations

4.1 Significant of outcomes

In conclusion, the application of machine learning (ML) techniques for lung cancer analysis, prediction, classification, and detection holds immense promise in revolutionizing the landscape of lung cancer care. Through the synthesis of medical imaging data, genomic information, and clinical variables, ML algorithms have demonstrated remarkable capabilities in improving early detection rates, enhancing diagnostic accuracy, guiding personalized treatment strategies, and predicting patient outcomes. The advancements in ML-driven approaches have the potential to significantly impact patient care by facilitating timely interventions, optimizing resource allocation, and ultimately improving survival rates for individuals diagnosed with lung cancer.

Future Recommendations and Significance of Outcomes:

i. **Clinical Translation and Implementation:** The successful translation of ML-based solutions from research laboratories to clinical practice is paramount. Future research should focus on validating the real-world performance of these algorithms in diverse patient populations and healthcare settings through prospective clinical trials and implementation studies. Collaboration with healthcare providers, regulatory agencies, and policymakers is essential to streamline the integration of ML-driven tools into routine clinical workflows and ensure their adoption by clinicians.

ii. **Interpretability and Transparency:** Enhancing the interpretability and transparency of ML models is critical for fostering trust among healthcare providers and patients. Future research should explore novel methods for explaining model predictions, elucidating the underlying decision-making processes, and providing actionable insights that clinicians can use to inform patient care. Developing standardized guidelines and best practices for reporting and evaluating the interpretability of ML algorithms can further enhance their clinical utility and acceptance.

iii. **Data Quality and Generalization:** Addressing issues related to data quality, bias, and generalization is essential for ensuring the robustness and reliability of ML models in lung cancer analysis. Future efforts should focus on expanding the availability of annotated training datasets representative of diverse patient demographics, disease subtypes, and clinical scenarios. Collaborative initiatives involving multi-institutional data sharing, data harmonization, and federated learning can facilitate the development of more generalizable and equitable ML solutions for lung cancer care.

iv. **Ethical and Societal Implications:** As ML technologies continue to advance, it is imperative to consider their ethical, legal, and societal implications. Future research should explore ethical frameworks for the responsible deployment of ML-driven tools in lung cancer diagnosis and management, addressing issues such as patient privacy, data security, informed consent, and algorithmic bias. Engaging with stakeholders, including patients, advocacy groups, and policymakers, can help ensure that ML technologies are developed and deployed in a manner that prioritizes patient well-being and societal benefit.

v. **Continued Innovation and Collaboration:** The field of ML in lung cancer analysis is rapidly evolving, driven by ongoing innovations in computational techniques, data science methodologies, and biomedical research. Future research should embrace interdisciplinary collaboration across domains such as computer science, medicine, biology, and ethics to harness the full potential of ML in improving lung cancer care. By fostering a culture of innovation, collaboration, and continuous improvement, we can collectively advance the field and make significant strides towards combating lung cancer and improving patient outcomes worldwide.

In summary, the outcomes of this thesis underscore the transformative potential of machine learning in lung cancer analysis and highlight the importance of interdisciplinary collaboration, ethical considerations, and real-world validation in translating research findings into meaningful clinical impact. By leveraging the power of machine learning technologies and embracing a patient-centered approach, we can pave the way towards a future where lung cancer is detected earlier, treated more effectively, and ultimately conquered.

4.2 Recommendation for Future Work

The utilization of machine learning (ML) in lung cancer analysis, prediction, classification, and detection represents a significant advancement in the fight against this pervasive disease. By leveraging diverse datasets encompassing medical imaging, genomic profiles, clinical data, and more, ML algorithms have demonstrated the potential to enhance early detection rates, refine diagnostic accuracy, personalize treatment strategies, and improve prognostic predictions. These advancements hold promise for revolutionizing lung cancer care, ultimately leading to better patient outcomes and reduced mortality rates.

Throughout this thesis, we have explored various ML techniques and their applications in lung cancer analysis, acknowledging both the progress made and the challenges encountered. While ML-based approaches have shown great promise, several areas warrant further investigation and development to maximize their clinical impact and ensure their responsible integration into healthcare systems.

Recommendations for Future Work:

i. **Enhanced Data Collection and Integration:** Future research efforts should prioritize the collection and integration of comprehensive and diverse datasets, including longitudinal patient data, multi-omics profiles, and real-world clinical outcomes. Collaborative initiatives and data-sharing agreements can facilitate access to large-scale datasets, enabling the development of more robust and generalizable ML models for lung cancer analysis.

ii. **Integration of Novel Data Sources:** Exploring the integration of emerging data sources, such as wearable sensors, digital pathology images, and patient-reported

outcomes, can provide valuable insights into disease progression, treatment response, and quality of life. Integrating these data streams into ML frameworks can enrich predictive models and enable personalized healthcare interventions tailored to individual patient needs.

iii. **Interpretability and Explainability:** Addressing the interpretability and explainability of ML models is crucial for fostering clinician trust and facilitating their adoption in clinical practice. Future research should focus on developing transparent ML algorithms and visualization techniques that provide clinicians with actionable insights into model predictions, enabling informed decision-making and patient-centered care.

iv. **Clinical Validation and Translation:** Rigorous validation studies and prospective clinical trials are essential for assessing the clinical utility and real-world performance of ML-driven tools for lung cancer analysis. Collaborations with healthcare institutions, regulatory bodies, and industry partners can expedite the translation of research findings into clinical practice, ensuring seamless integration and scalability of ML-based solutions.

v. **Ethical Considerations and Regulatory Compliance:** Ethical considerations, including patient privacy, data security, informed consent, and algorithmic bias, must be carefully addressed throughout the development and deployment of ML technologies in lung cancer care. Adhering to ethical guidelines and regulatory standards is imperative to ensure patient safety, privacy, and autonomy in the use of ML-driven healthcare technologies.

By addressing these recommendations for future work, researchers can advance the field of ML in lung cancer analysis and contribute to the ongoing efforts to improve patient care and outcomes in lung cancer management.

In summary, while ML holds tremendous potential for transforming lung cancer diagnosis, prediction, classification, and detection, continued research, collaboration, and innovation are essential to realize this potential fully. By embracing these recommendations and working together across disciplines, we can drive forward progress in lung cancer research and ultimately make a meaningful difference in the lives of patients affected by this devastating disease.

5. Appendixes:

Import Libraries:

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

Visualize Data:

```
data = pd.read_csv('survey_lung_cancer.csv', header=0, sep=",")
print("Visualize Data : \n", data)
```

Basic Exploration:

```
DatasetShape = data.shape;
print("Shape of Dataset")
print(f"Number of Row : {DatasetShape[0]}")
print(f"Number of Column : {DatasetShape[1]}")
```

About Dataset Information:

```
print(f"Informations about the  Dataset :\n ", data.info())
```

Output :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   GENDER                 309 non-null    object
 1   AGE                    309 non-null    int64
 2   SMOKING                309 non-null    int64
 3   YELLOW_FINGERS         309 non-null    int64
 4   ANXIETY                309 non-null    int64
 5   PEER_PRESSURE          309 non-null    int64
 6   CHRONIC DISEASE        309 non-null    int64
 7   FATIGUE                309 non-null    int64
 8   ALLERGY                309 non-null    int64
 9   WHEEZING               309 non-null    int64
 10  ALCOHOL CONSUMING      309 non-null    int64
 11  COUGHING               309 non-null    int64
 12  SHORTNESS OF BREATH    309 non-null    int64
 13  SWALLOWING DIFFICULTY  309 non-null    int64
 14  CHEST PAIN             309 non-null    int64
 15  LUNG_CANCER            309 non-null    object
dtypes: int64(14), object(2)
memory usage: 38.8+ KB
Informations about the  Dataset :
  None
```

Summary of Dataset:

```
x = data['AGE']
print("Mean of Age : ", np.mean(x))
print("STD : ", np.std(x))
print("Min : ", np.min(x))
print("Max : ", np.max(x))
print("25% : ", np.percentile(x,25))
```

```
# Find Duplicate Data,
duplicateDataFind = data[data.duplicated()].shape[0]
print(f"The Duplicate Data found : {duplicateDataFind} entries amon
{data.shape[0]} entries in this dataset")

# Remove duplication data,
data.drop_duplicates(keep='first', inplace=True)
print(f"*After dropping/remove duplicate entries there are
{data.shape[0]} entries in this dataset")
```

Data Processing:

```
import pandas as pd

data_temp = data.copy()

# Replace gender code with coresponding labels,
data_temp["GENDER"].replace({"M": "Male", "F":"Female"},inplace
=True)

# Replace Numerical Values 2 = "Yes", 1 = "No",
data_temp.replace({2:"Yes",1:"No"},inplace=True)

# Display some modifiyed Data :
data_temp.head()

# here show just only the lung cancer positive data,
data_temp_pos = data_temp[data_temp["LUNG_CANCER"]=="YES"]
data_temp_pos.head()
```

Age Distribution:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set seaborn style and context
```

```
sns.set_style("whitegrid")
sns.set_context("poster", font_scale=0.7)

# Define palette
palette = ["#1d7874", "#679289", "#f4c095", "#ee2e31", "#ffb563",
"#918450", "#f85e00", "#a41623", "#9a031e", "#d6d6d6", "#ffee32",
"#ffd100", "#333533", "#202020"]

# Plot color palette
sns.palplot(sns.color_palette(palette))

# Show plot
plt.show()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create subplots with shared axes
fig, axs = plt.subplots(2, 1, figsize=(20, 10), sharex=True,
sharey=True)
fig.subplots_adjust(hspace=0.5)

# Generate color palette
palette = sns.color_palette("husl", 5)

# Define plot titles
titles = ["Positive Male Cases Age Distribution", "Positive Female
Cases Age Distribution"]

# Plot histograms for male and female cases age distribution
for gender, ax, title, color in zip(["Male", "Female"], axs, titles,
palette):
    sns.histplot(data_temp_pos[data_temp_pos["GENDER"] ==
gender]["AGE"],
                 color=color, kde=True, ax=ax, bins=20, alpha=0.7)
    ax.set_title(title, fontsize=20, pad=20)
    ax.set_xlabel("Age", fontsize=14)
    ax.set_ylabel("Quantity", fontsize=14)

    # Remove spines
    sns.despine(ax=ax, left=True, bottom=True)

plt.show()
```

```
import matplotlib.pyplot as plt

# Data
gender_counts =
[len(data_temp_pos[data_temp_pos["GENDER"]=="Male"]["GENDER"]),
                 len(data_temp_pos[data_temp_pos["GENDER"]=="Female"]
["GENDER"])]
labels = "Male", "Female"
colors = ["#0f4c5c", "#FFC300"]

# Create figure and axes
fig, ax = plt.subplots(figsize=(8,8))

# Plot pie chart
wedges, texts, autotexts = ax.pie(gender_counts,
                                  explode=(0, 0),
                                  labels=labels,
                                  colors=colors,
                                  autopct="%.2f%%",
                                  pctdistance=0.7,
                                  textprops=dict(size=25,
color="white"),
                                  wedgeprops=dict(width=0.5,
edgecolor="white", linewidth=5),
                                  shadow=True,
                                  startangle=0)

# Add legend
ax.legend(wedges, labels, title="Category", loc="center left",
bbox_to_anchor=(1, 0, 0.5, 1))

# Add title
ax.set_title("\nPositive Cases' Gender Distribution", fontsize=20)

# Show plot
plt.show()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the figure and axes
```

```
fig, axs = plt.subplots(1, 2, figsize=(20, 8), sharex=True,
sharey=True)

# Plot countplots for smoking and alcohol consumption
for ax, column in zip(axs, ["SMOKING", "ALCOHOL CONSUMING"]):
    sns.countplot(data=data_temp_pos, x="GENDER", hue=column,
hue_order=["Yes", "No"], palette=["#0f4c5c", "#FFC300"],
saturation=1, ax=ax)
    ax.set_title(f"\nEffect of {column.title()}\n", fontsize=20)
    ax.set_xlabel("Gender")
    ax.set_ylabel("Quantity")
    for container in ax.containers:
        ax.bar_label(container, label_type="center", padding=2,
size=25, color="white", rotation=0)

# Improve layout and remove spines
plt.tight_layout(pad=4.0)
sns.despine(left=True, bottom=True)

# Show plot
plt.show()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the figure and axes
fig, axs = plt.subplots(2, 5, figsize=(20, 14), sharex=False,
sharey=True)
plt.tight_layout(pad=4.0)

# List of columns to plot
columns_to_plot = ["YELLOW_FINGERS", "ANXIETY", "CHRONIC DISEASE",
"CHEST PAIN", "FATIGUE ", "WHEEZING", "COUGHING", "SHORTNESS OF
BREATH", "SWALLOWING DIFFICULTY", "ALLERGY "]

# Plot countplots for each column
for ax, column in zip(axs.flatten(), columns_to_plot):
    sns.countplot(data=data_temp_pos, x="GENDER", hue=column,
hue_order=["Yes", "No"], palette=["#0f4c5c", "#FFC300"],
saturation=1, ax=ax)
    ax.set_ylabel("Total")
    ax.legend(title=column.title(), loc="upper right")
    for container in ax.containers:
        ax.bar_label(container, label_type="center", padding=2,
size=17, color="white", rotation=0)
```

```
# Remove spines
sns.despine(left=True, bottom=True)

# Show plot
plt.show()
```

```
import pandas as pd

# Replace gender codes with corresponding labels
data["GENDER"] = data["GENDER"].replace({"M": "Male", "F": "Female"})

# Encode the target variable using LabelEncoder
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data["LUNG_CANCER"] =
label_encoder.fit_transform(data["LUNG_CANCER"])

# Convert categorical variable 'GENDER' into dummy/indicator
variables
data = pd.get_dummies(data, columns=["GENDER"], drop_first=True)
data.rename(columns={"GENDER_Male": "MALE", "GENDER_Female":
"FEMALE"}, inplace=True)

data.head()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the figure size
plt.figure(figsize=(16, 12))

# Plot heatmap
sns.heatmap(data.corr(), cmap="summer", square=True,
cbar_kws={"shrink": 0.99},
            annot=True, vmin=-1, vmax=1, linewidths=0.1,
linecolor='white', annot_kws={"fontsize": 12})

# Set title and rotate x-axis labels
plt.title("Pearson Correlation Of Features\n", size=15)
plt.xticks(rotation=90)

# Show plot
plt.show()
```

Data Pre-processing & Classification:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Split the data into features (x) and target (y)
x = data.drop("LUNG_CANCER", axis=1)
y = data["LUNG_CANCER"]

# Perform train-test split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

# Scale the features using StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Print the shapes of training and testing data
print(f"Shape of training data: {x_train_scaled.shape},
{y_train.shape}")
print(f"Shape of testing data: {x_test_scaled.shape},
{y_test.shape}")
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Create a pipeline with StandardScaler and LogisticRegression
lr_pipeline = make_pipeline(StandardScaler(), LogisticRegression())

# Fit the model and make predictions
lr_pipeline.fit(x_train, y_train)
lr_pred = lr_pipeline.predict(x_test)

# Calculate evaluation metrics
lr_conf = confusion_matrix(y_test, lr_pred)
lr_report = classification_report(y_test, lr_pred)
lr_acc = round(accuracy_score(y_test, lr_pred) * 100, ndigits=2)

# Print evaluation results
print(f"Confusion Matrix:\n{lr_conf}\n")
```

```
print(f"Classification Report:\n{lr_report}\n")
print(f"The Accuracy of Logistic Regression is {lr_acc}%")
print(f"Predict Data : {lr_pred}")
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Create a pipeline with StandardScaler and DecisionTreeClassifier
dt_pipeline = make_pipeline(StandardScaler(),
DecisionTreeClassifier())

# Fit the model and make predictions
dt_pipeline.fit(x_train, y_train)
dt_pred = dt_pipeline.predict(x_test)

# Calculate evaluation metrics
dt_conf = confusion_matrix(y_test, dt_pred)
dt_report = classification_report(y_test, dt_pred)
dt_acc = round(accuracy_score(y_test, dt_pred) * 100, ndigits=2)

# Print evaluation results
print(f"Confusion Matrix:\n{dt_conf}\n")
print(f"Classification Report:\n{dt_report}\n")
print(f"The Accuracy of Decision Tree Classifier is {dt_acc}%")
print(f"Predict Data : {dt_pred}")
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Create a pipeline with StandardScaler and GaussianNB
nb_pipeline = make_pipeline(StandardScaler(), GaussianNB())

# Fit the model and make predictions
nb_pipeline.fit(x_train, y_train)
nb_pred = nb_pipeline.predict(x_test)

# Calculate evaluation metrics
nb_conf = confusion_matrix(y_test, nb_pred)
```

```
nb_report = classification_report(y_test, nb_pred)
nb_acc = round(accuracy_score(y_test, nb_pred) * 100, ndigits=2)

# Print evaluation results
print(f"Confusion Matrix:\n{nb_conf}\n")
print(f"Classification Report:\n{nb_report}\n")
print(f"The Accuracy of Naive Bayes Classifier is {nb_acc}%")
print(f"Predict Data : ", nb_pred)
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Create a pipeline with StandardScaler and MLPClassifier
nn_pipeline = make_pipeline(StandardScaler(),
MLPClassifier(hidden_layer_sizes=(100, ), max_iter=1000))

# Fit the model and make predictions
nn_pipeline.fit(x_train, y_train)
nn_pred = nn_pipeline.predict(x_test)

# Calculate evaluation metrics
nn_conf = confusion_matrix(y_test, nn_pred)
nn_report = classification_report(y_test, nn_pred)
nn_acc = round(accuracy_score(y_test, nn_pred) * 100, ndigits=2)

# Print evaluation results
print(f"Confusion Matrix:\n{nn_conf}\n")
print(f"Classification Report:\n{nn_report}\n")
print(f"The Accuracy of Neural Network Classifier is {nn_acc}%")
print(f"Predict Data : {nn_pred}")
```

Neural Network Architecture:

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

```
from tensorflow.keras import regularizers

regularization_parameter = 0.003

# Define the neural network model
neural_model = Sequential([
    Dense(units=32, input_dim=x_train.shape[-1], activation="relu",
kernel_regularizer=regularizers.l1(regularization_parameter)),
    Dense(units=64, activation="relu",
kernel_regularizer=regularizers.l1(regularization_parameter)),
    Dense(units=128, activation="relu",
kernel_regularizer=regularizers.l1(regularization_parameter)),
    Dense(units=16, activation="relu",
kernel_regularizer=regularizers.l1(regularization_parameter)),
    Dense(units=1, activation="sigmoid")
])

# Print model summary
print(neural_model.summary())
```

```
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
import numpy as np

# Convert input data to numpy arrays
x_train = np.array(x_train).astype(np.float32)
y_train = np.array(y_train).astype(np.float32)
x_test = np.array(x_test).astype(np.float32)
y_test = np.array(y_test).astype(np.float32)

# Define EarlyStopping callback
early_stopping = EarlyStopping(monitor='accuracy', patience=0,
mode='max', verbose=1)

# Define neural network model
neural_model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=32, input_dim=x_train.shape[-1],
activation='relu'),
    tf.keras.layers.Dense(units=64, activation='relu'),
    tf.keras.layers.Dense(units=128, activation='relu'),
    tf.keras.layers.Dense(units=16, activation='relu'),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
```

```
])

# Compile the neural network model
neural_model.compile(optimizer=Adam(learning_rate=0.001),
                     loss='binary_crossentropy',
                     metrics=['accuracy'])

# Train the model with EarlyStopping callback
history = neural_model.fit(x_train, y_train,
                           epochs=150,
                           verbose=1,
                           batch_size=64,
                           validation_data=(x_test, y_test),
                           callbacks=[early_stopping])
```

```
import matplotlib.pyplot as plt

# Extracting metrics from history
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]

# Number of epochs
epochs = range(len(acc))

# Plotting
plt.figure(figsize=(14, 10))

# Training and validation accuracy
plt.subplot(2, 1, 1)
plt.plot(epochs, acc, color='blue', marker='o', linestyle='-',
label='Training Accuracy')
plt.plot(epochs, val_acc, color='red', marker='s', linestyle='--',
label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Training and validation loss
plt.subplot(2, 1, 2)
plt.plot(epochs, loss, color='blue', marker='o', linestyle='-',
label='Training Loss')
```

```
plt.plot(epochs, val_loss, color='red', marker='s', linestyle='--',
label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Show plot
plt.tight_layout()
plt.show()
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

# Splitting the data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

# Creating and training the Random Forest Model
rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)

# Evaluating Random Forest Model
rf_accuracy = rf_model.score(x_test, y_test)
print("Random Forest Accuracy:", rf_accuracy)

# Extracting feature importances
feature_importances = rf_model.feature_importances_

# Sorting the feature importances in descending order
sorted_idx = feature_importances.argsort()

# Plotting feature importances
plt.figure(figsize=(10, 8))
plt.barh(range(len(sorted_idx)), feature_importances[sorted_idx],
align='center')
plt.yticks(range(len(sorted_idx)), [x.columns[i] for i in
sorted_idx])
plt.title('Feature Importances in Random Forest Model')
plt.xlabel('Relative Importance')
plt.ylabel('Feature')
plt.show()
```