CHAPTER I - INTRODUCTION

1.1 Scope of analysis:

Lung cancer is a significant public health concern worldwide, representing one of the leading causes of cancer-related morbidity and mortality. Understanding the multifactorial nature of this disease is crucial for both prevention and effective management. To this end, datasets containing comprehensive information about patients with lung cancer play a pivotal role in advancing research and informing healthcare strategies.

This dataset compiles detailed information on various factors potentially associated with lung cancer incidence and progression. It includes demographic data such as age and gender, as well as lifestyle factors like alcohol use, smoking habits (both active and passive), and dietary patterns. Environmental exposures, including air pollution and occupational hazards, are also documented. Furthermore, medical history elements such as chronic lung disease, genetic predisposition, and symptoms like chest pain, coughing of blood, fatigue, and weight loss, are included.

Additionally, the dataset encompasses clinical manifestations such as shortness of breath, wheezing, swallowing difficulty, and clubbing of finger nails, which may provide insights into disease severity and progression. Moreover, sleep-related factors like snoring are also considered, recognizing the potential impact of sleep-disordered breathing on lung health.

By analyzing this comprehensive dataset, researchers and healthcare professionals can identify potential risk factors, elucidate underlying mechanisms, and develop targeted interventions to mitigate the burden of lung cancer. Additionally, this dataset facilitates the development of predictive models to enhance early detection and personalized treatment approaches, ultimately improving patient outcomes and reducing the global burden of lung cancer.

1.2 Objective:

The primary objective of analyzing the dataset on lung cancer patients is to identify and understand the various risk factors associated with the development and progression of the disease. In this project various machine learning algorithms is used to train the data and predict the possible outcomes using python language. python is a popular data analysis language because of its extensive libraries for data processing, visualization, machine learning, and a variety of other tasks.

1.3 Data Collection:

Data collection is the process of gathering and measuring information on targeted variables of interest in an organized system, which then allows you to answer the relevant questions. A public dataset available from **Kaggle**, the actual source of the data is not mention as it confidential. No matter where you obtain your data from, make sure your data is Relevant and Validated. Quality is the key!

CHAPTER II - DATA UNDERSTANDING

2.1 Data understanding

The dataset encompasses demographic data (age, gender), lifestyle factors (alcohol, smoking), and environmental exposures (pollution, occupational hazards). Medical history includes chronic lung disease, genetic risk, and symptoms like chest pain, coughing of blood, and fatigue. Clinical manifestations such as shortness of breath, wheezing, and clubbing of finger nails are recorded. Sleep-related factors like snoring are also considered. The dataset aims to identify risk factors associated with lung cancer incidence and progression. It facilitates predictive modeling for early detection and personalized treatment approaches. Insights from the dataset inform public health interventions targeting modifiable risk factors like smoking and pollution exposure. Understanding disparities in lung cancer incidence across demographics guides efforts toward healthcare equity. Overall, the dataset aids in advancing research, improving clinical decision-making, and reducing the global burden of lung cancer.

Importing the libraries

1. Pandas – for reading the dataset files
2. NumPy – for numerical calculations
3. Matplotlib – for graphic visualization
4. Seaborn – for graphical visualization of the data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")
sns.set_palette("RdBu")
```

Reading Dataset

Read the dataset using the pandas library.

```
#Loading the dataset
df = pd.read_csv("E:/project/cancer patient data sets.csv")
df
```

1.2s Python

	index	Patient Id	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	...	Fatigue	Weight Loss	Shortness of Breath	Wheezing	Swallowing Difficulty	Clubbing of Finger Nails	Frequent Cold	Dry Cough	Snoring	Level
0	0	P1	33	1	2	4	5	4	3	2	...	3	4	2	2	3	1	2	3	4	Low
1	1	P10	17	1	3	1	5	3	4	2	...	1	3	7	8	6	2	1	7	2	Medium
2	2	P100	35	1	4	5	6	5	5	4	...	8	7	9	2	1	4	6	7	2	High
3	3	P1000	37	1	7	7	7	7	6	7	...	4	2	3	1	4	5	6	7	5	High
4	4	P101	46	1	6	8	7	7	7	6	...	3	2	4	1	4	2	4	2	3	High
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	995	P995	44	1	6	7	7	7	7	6	...	5	3	2	7	8	2	4	5	3	High
996	996	P996	37	2	6	8	7	7	7	6	...	9	6	5	7	2	4	3	1	4	High
997	997	P997	25	2	4	5	6	5	5	4	...	8	7	9	2	1	4	6	7	2	High
998	998	P998	18	2	6	8	7	7	7	6	...	3	2	4	1	4	2	4	2	3	High
999	999	P999	47	1	6	5	6	5	5	4	...	8	7	9	2	1	4	6	7	2	High

Shape of the dataset

Check the shape of the dataset **data. Shape ()** command

```
df.shape
```

✓ 0.0s

```
(1000, 26)
```

This dataset contains 10000 records and 26 columns

2.2 Data Description

A variable consists of two parts – the label and the data type. Data types can be numeric (integers, real numbers) or strings. The data type can sometimes be tricky; for example, US postal codes are numeric but need to be treated as strings. Once the labels and data types are known, you can group attributes into two kinds for modeling:

Continuous Variables: These are numbers which can range from negative infinity to positive infinity. You would associate with the labels a sense of magnitude, maximum and minimum. You can sort on such variables and filter by ranges.

Categorical Variables: These variables can have a limited set of values, each of which indicate a sub-type. For example, Direction is a categorical variable because it can be either North, South, East, or West. You can filter on or group by a specific value or values of a categorical variable

Now, let's look into the variables of our dataset. Once you have identified the variables of interest, summary statistics help you understand the nature of each variable. Each attribute's summary statistics such as **count, standard deviation, mean, minimum, maximum and IQR values** are calculated using the describe() function. To dive into the dataset, Python Programming is used for further process.

Here **describe()** function is used in python to derive the overall summary of the dataset. But in the dataset most of the values are categories. Therefore all datatypes are included in the function as describe() function only takes numeric datatype as default.

```
df.describe()
```

✓ 0.5s Python

	index	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	Balanced Diet	...	Coughing of Blood	Fatigue	Weight Loss	Shortness of Breath	Wheezing	Swa D
count	1000.000000	1000.000000	1000.000000	1000.0000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	...	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	100
mean	499.500000	37.174000	1.402000	3.8400	4.563000	5.165000	4.840000	4.580000	4.380000	4.491000	...	4.859000	3.856000	3.855000	4.240000	3.777000	
std	288.819436	12.005493	0.490547	2.0304	2.620477	1.980833	2.107805	2.126999	1.848518	2.135528	...	2.427965	2.244616	2.206546	2.285087	2.041921	
min	0.000000	14.000000	1.000000	1.0000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	
25%	249.750000	27.750000	1.000000	2.0000	2.000000	4.000000	3.000000	2.000000	3.000000	2.000000	...	3.000000	2.000000	2.000000	2.000000	2.000000	
50%	499.500000	36.000000	1.000000	3.0000	5.000000	6.000000	5.000000	5.000000	4.000000	4.000000	...	4.000000	3.000000	3.000000	4.000000	4.000000	
75%	749.250000	45.000000	2.000000	6.0000	7.000000	7.000000	7.000000	7.000000	6.000000	7.000000	...	7.000000	5.000000	6.000000	6.000000	5.000000	
max	999.000000	73.000000	2.000000	8.0000	8.000000	8.000000	8.000000	7.000000	7.000000	7.000000	...	9.000000	9.000000	8.000000	9.000000	8.000000	

Summarizing each attribute

Let’s get to know about each and every variable below using describe () function individually.

Patient ID:

```
df['Patient Id'].describe ()
```

✓ 0.1s

```
count     1000
unique    1000
top         P1
freq         1
Name: Patient Id, dtype: object
```

Age:

```
df['Age'].describe()
```

✓ 0.2s

```
count    1000.000000
mean       37.174000
std        12.005493
min        14.000000
25%        27.750000
50%        36.000000
75%        45.000000
max        73.000000
Name: Age, dtype: float64
```

Gender:

```
df['Gender'].describe()
```

✓ 0.3s

```
count    1000.000000
mean        1.402000
std         0.490547
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max         2.000000
Name: Gender, dtype: float64
```

Air pollution:

```
df['Air Pollution'].describe()
```
✓ 0.4s

```
count    1000.0000
mean        3.8400
std         2.0304
min         1.0000
25%         2.0000
50%         3.0000
75%         6.0000
max         8.0000
Name: Air Pollution, dtype: float64
```

Alcohol Use:

```
df['Air Pollution'].describe()
```
✓ 0.4s

```
count    1000.0000
mean        3.8400
std         2.0304
min         1.0000
25%         2.0000
50%         3.0000
75%         6.0000
max         8.0000
Name: Air Pollution, dtype: float64
```

Dust allergy:

```
df['Dust Allergy'].describe()
```
✓ 0.3s

```
count    1000.000000
mean        5.165000
std         1.980833
min         1.000000
25%         4.000000
50%         6.000000
75%         7.000000
max         8.000000
Name: Dust Allergy, dtype: float64
```

Occupational Hazards:

```
df['OccuPational Hazards'].describe()
```
✓ 0.2s

```
count    1000.000000
mean        4.840000
std         2.107805
min         1.000000
25%         3.000000
50%         5.000000
75%         7.000000
max         8.000000
Name: OccuPational Hazards, dtype: float64
```

Genetic Risk:

```
df['Genetic Risk'].describe()
```
✓ 0.3s

```
count    1000.000000
mean        4.580000
std         2.126999
min         1.000000
25%         2.000000
50%         5.000000
75%         7.000000
max         7.000000
Name: Genetic Risk, dtype: float64
```

Chronic Lung disease:

```
df['chronic Lung Disease'].describe()
```
✓ 0.5s

```
count    1000.000000
mean        4.380000
std         1.848518
min         1.000000
25%         3.000000
50%         4.000000
75%         6.000000
max         7.000000
Name: chronic Lung Disease, dtype: float64
```

Balanced diet:

```
df['Balanced Diet'].describe()
```
✓ 0.4s
```
count    1000.000000
mean        4.491000
std         2.135528
min         1.000000
25%         2.000000
50%         4.000000
75%         7.000000
max         7.000000
Name: Balanced Diet, dtype: float64
```

Obesity:

```
df['Obesity'].describe()
```
✓ 0.5s
```
count    1000.000000
mean        4.465000
std         2.124921
min         1.000000
25%         3.000000
50%         4.000000
75%         7.000000
max         7.000000
Name: Obesity, dtype: float64
```

Smoking:

```
df['Smoking'].describe()
```
✓ 0.3s
```
count    1000.000000
mean        3.948000
std         2.495902
min         1.000000
25%         2.000000
50%         3.000000
75%         7.000000
max         8.000000
Name: Smoking, dtype: float64
```

Passive smoker:

```
df['Passive Smoker'].describe()
```
✓ 0.3s

```
count    1000.000000
mean        4.195000
std         2.311778
min         1.000000
25%         2.000000
50%         4.000000
75%         7.000000
max         8.000000
Name: Passive Smoker, dtype: float64
```

Chest pain:

```
df['Chest Pain'].describe()
```
✓ 0.3s

```
count    1000.000000
mean        4.438000
std         2.280209
min         1.000000
25%         2.000000
50%         4.000000
75%         7.000000
max         9.000000
Name: Chest Pain, dtype: float64
```

Coughing of blood:

```
df['Coughing of Blood'].describe()
```
✓ 0.2s

```
count    1000.000000
mean        4.859000
std         2.427965
min         1.000000
25%         3.000000
50%         4.000000
75%         7.000000
max         9.000000
Name: Coughing of Blood, dtype: float64
```

Fatigue:

```
df['Fatigue'].describe()
```
✓ 0.3s

```
count    1000.000000
mean        3.856000
std         2.244616
min         1.000000
25%         2.000000
50%         3.000000
75%         5.000000
max         9.000000
Name: Fatigue, dtype: float64
```

Weight loss:

```
df['Weight Loss'].describe()
```
✓ 0.3s

```
count    1000.000000
mean        3.855000
std         2.206546
min         1.000000
25%         2.000000
50%         3.000000
75%         6.000000
max         8.000000
Name: Weight Loss, dtype: float64
```

Shortness of breath:

```
df['Shortness of Breath'].describe()
```
✓ 0.7s

```
count    1000.000000
mean        4.240000
std         2.285087
min         1.000000
25%         2.000000
50%         4.000000
75%         6.000000
max         9.000000
Name: Shortness of Breath, dtype: float64
```

Wheezing:

```
df['Wheezing'].describe()
```
✓ 0.2s

```
count    1000.000000
mean        3.777000
std         2.041921
min         1.000000
25%         2.000000
50%         4.000000
75%         5.000000
max         8.000000
Name: Wheezing, dtype: float64
```

Swallowing Difficulty:

```
df['Swallowing Difficulty'].describe()
```
✓ 0.2s

```
count    1000.000000
mean        3.746000
std         2.270383
min         1.000000
25%         2.000000
50%         4.000000
75%         5.000000
max         8.000000
Name: Swallowing Difficulty, dtype: float64
```

Clubbing of finger nails:

```
df['Clubbing of Finger Nails'].describe()
```
✓ 0.4s

```
count    1000.000000
mean        3.923000
std         2.388048
min         1.000000
25%         2.000000
50%         4.000000
75%         5.000000
max         9.000000
Name: Clubbing of Finger Nails, dtype: float64
```

Frequent cold:

```
df['Frequent Cold'].describe()
```
✓ 0.3s
```
count    1000.000000
mean        3.536000
std         1.832502
min         1.000000
25%         2.000000
50%         3.000000
75%         5.000000
max         7.000000
Name: Frequent Cold, dtype: float64
```

Dry cough:

```
df['Dry Cough'].describe()
```
✓ 0.3s
```
count    1000.000000
mean        3.853000
std         2.039007
min         1.000000
25%         2.000000
50%         4.000000
75%         6.000000
max         7.000000
Name: Dry Cough, dtype: float64
```

Snoring:

```
df['Snoring'].describe()
```
✓ 0.3s
```
count    1000.000000
mean        2.926000
std         1.474686
min         1.000000
25%         2.000000
50%         3.000000
75%         4.000000
max         7.000000
Name: Snoring, dtype: float64
```

Level:

```
df['Level'].describe()
```

✓ 0.4s

```
count     1000
unique       3
top       High
freq       365
Name: Level, dtype: object
```

CHAPTER III- DATA PREPARATION

3.1 Data Cleaning

Around 80% of time will be spent cleaning data. Cleaning your data is a process of ensuring that the data is in the correct format; consistent and errors are identified and dealt with appropriately. The actions below lead to a cleaner dataset:

- Remove duplicate values (This is usually the case when combining multiple datasets)
- Remove irrelevant observations (observations need to be specific to the problem you are solving)
- Address missing values (e.g., Imputation techniques, drop features/observations)
- Reformat data types (e.g., Boolean, numeric, Date time)
- Filter unwanted outliers (if you have a legitimate reason)
- Reformat strings (e.g., remove white spaces, mislabeled /misspelt categories)
- Validate (does the data make sense? does the data adhere to the defined business rules?)

Cleaning your data will allow for higher-quality information and ultimately lead to more conclusive and accurate decision.

Before getting into this step, Let's take look of the overall summary of the dataset using **info()** function in python.

```
df.info()
```
✓ 0.9s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   index                     1000 non-null   int64
 1   Patient Id                1000 non-null   object
 2   Age                       1000 non-null   int64
 3   Gender                    1000 non-null   int64
 4   Air Pollution             1000 non-null   int64
 5   Alcohol use               1000 non-null   int64
 6   Dust Allergy              1000 non-null   int64
 7   OccuPational Hazards      1000 non-null   int64
 8   Genetic Risk              1000 non-null   int64
 9   chronic Lung Disease      1000 non-null   int64
 10  Balanced Diet             1000 non-null   int64
 11  Obesity                   1000 non-null   int64
 12  Smoking                   1000 non-null   int64
 13  Passive Smoker            1000 non-null   int64
 14  Chest Pain                1000 non-null   int64
 15  Coughing of Blood         1000 non-null   int64
 16  Fatigue                   1000 non-null   int64
 17  Weight Loss               1000 non-null   int64
 18  Shortness of Breath       1000 non-null   int64
 19  Wheezing                  1000 non-null   int64
 20  Swallowing Difficulty     1000 non-null   int64
 21  Clubbing of Finger Nails  1000 non-null   int64
 22  Frequent Cold             1000 non-null   int64
 23  Dry Cough                 1000 non-null   int64
 24  Snoring                   1000 non-null   int64
 25  Level                     1000 non-null   object
dtypes: int64(24), object(2)
memory usage: 203.2+ KB
```

3.2 Handling null and NA values

Identifying the null values in the dataset

```
df.isna().sum()
```
✓ 0.1s

```
index                     0
Patient Id                0
Age                       0
Gender                    0
Air Pollution             0
Alcohol use               0
Dust Allergy              0
OccuPational Hazards      0
Genetic Risk              0
chronic Lung Disease      0
Balanced Diet             0
Obesity                   0
Smoking                   0
Passive Smoker            0
Chest Pain                0
Coughing of Blood         0
Fatigue                   0
Weight Loss               0
Shortness of Breath       0
Wheezing                  0
Swallowing Difficulty     0
Clubbing of Finger Nails  0
Frequent Cold             0
Dry Cough                 0
Snoring                   0
Level                     0
dtype: int64
```

There are no null values in our dataset.

3.3 Cleaning Header Spaces

Variable names are case sensitive. As it should not contain any spaces in it. It should only consist of characters, numbers and under-scores. Now let clean the column spaces.

```
df.columns=df.columns.str.replace(" ","_")
df.columns
```

✓ 0.5s

```
Index(['index', 'Patient_Id', 'Age', 'Gender', 'Air_Pollution', 'Alcohol_use',
       'Dust_Allergy', 'OccuPational_Hazards', 'Genetic_Risk',
       'chronic_Lung_Disease', 'Balanced_Diet', 'Obesity', 'Smoking',
       'Passive_Smoker', 'Chest_Pain', 'Coughing_of_Blood', 'Fatigue',
       'Weight_Loss', 'Shortness_of_Breath', 'Wheezing',
       'Swallowing_Difficulty', 'Clubbing_of_Finger_Nails', 'Frequent_Cold',
       'Dry_Cough', 'Snoring', 'Level'],
      dtype='object')
```

3.4 Checking for duplicate values

Data replication requires keeping multiple copies of data, it can lead to higher storage costs. While these costs might not significantly affect large organizations, maintaining additional storage can negatively affect budgets for smaller organizations.

It causes issues such as,

Overfitting: Duplicate entries may artificially increase the weight or importance of certain observations in the training process, leading to overfitting.

Inflated accuracy: Duplicate entries can make the model appear more accurate than it actually is

In dataset, there is not duplicated values. So let’s move on to further steps.

```
df.duplicated().sum()
```

✓ 4.7s

```
0
```

CHAPTER IV – EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is the crucial process of using summary statistics and graphical representations to perform preliminary investigations on data in order to uncover patterns, detect anomalies, test hypotheses, and verify assumptions.

In simple words, EDA is a data exploration technique to understand the various aspects of the data. EDA is often used to see what data may disclose outside of formal modelling and to learn more about the variables in a data collection and how they interact. It could also help us figure out if the statistical procedures we are considering for data analysis are appropriate. Before modelling the data, it gives insight into all of the data and the numerous interactions between the data elements.

4.1 Data Visualization

Data visualization is extremely useful in understanding the data and obtaining useful insights. It can allow you to get an instant understanding of the data that is just not possible by observing rows of data in a table. That's what makes it so important in Data Science!

Let's see some more reasons why data visualization is so important.

Data Visualization discovers the trends in data. It is interactive and provides a perspective on the data. It explains a data process, tells a data story and puts the data into the correct context. Data visualization is educational for users and saves time.

Some charts below to understand the data visually.

Before starting the analysis, three new variables are created and stored some values in order to perform the analysis efficiently.

```
col=list(df.columns)
col
```

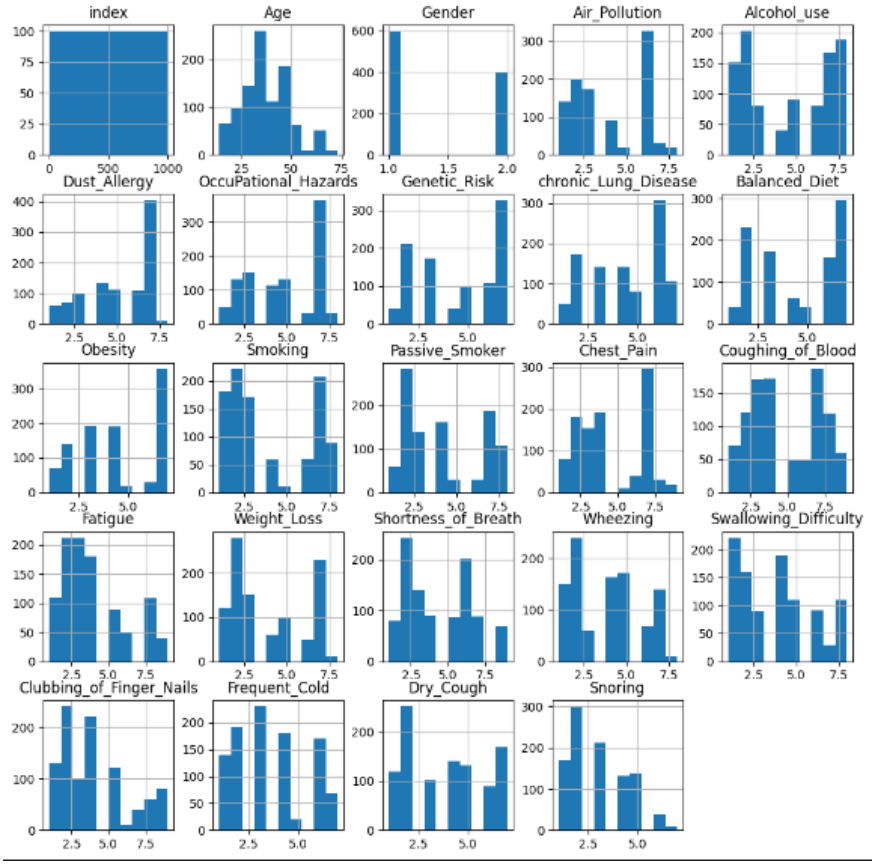
✓ 0.3s

```
['index',
 'Patient_Id',
 'Age',
 'Gender',
 'Air_Pollution',
 'Alcohol_use',
 'Dust_Allergy',
 'OccuPational_Hazards',
 'Genetic_Risk',
 'chronic_Lung_Disease',
 'Balanced_Diet',
 'Obesity',
 'Smoking',
 'Passive_Smoker',
 'Chest_Pain',
 'Coughing_of_Blood',
 'Fatigue',
 'Weight_Loss',
 'Shortness_of_Breath',
 'Wheezing',
 'Swallowing_Difficulty',
 'Clubbing_of_Finger_Nails',
 'Frequent_Cold',
 'Dry_Cough',
 'Snoring',
 'Level']
```

```
df.dtypes
```

✓ 0.5s

```
index                       int64
Patient_Id                 object
Age                         int64
Gender                      int64
Air_Pollution               int64
Alcohol_use                 int64
Dust_Allergy                int64
OccuPational_Hazards        int64
Genetic_Risk                int64
chronic_Lung_Disease        int64
Balanced_Diet               int64
Obesity                     int64
Smoking                     int64
Passive_Smoker              int64
Chest_Pain                  int64
Coughing_of_Blood           int64
Fatigue                     int64
Weight_Loss                 int64
Shortness_of_Breath         int64
Wheezing                    int64
Swallowing_Difficulty       int64
Clubbing_of_Finger_Nails    int64
Frequent_Cold               int64
Dry_Cough                   int64
Snoring                     int64
Level                      object
dtype: object
```



A pair plot visually displays pairwise relationships between variables in a dataset, often using scatterplots along the diagonal and scatterplots or histograms elsewhere. In the context of lung cancer data, it helps identify correlations between factors like smoking status, air pollution exposure, and symptoms, aiding in pattern recognition.

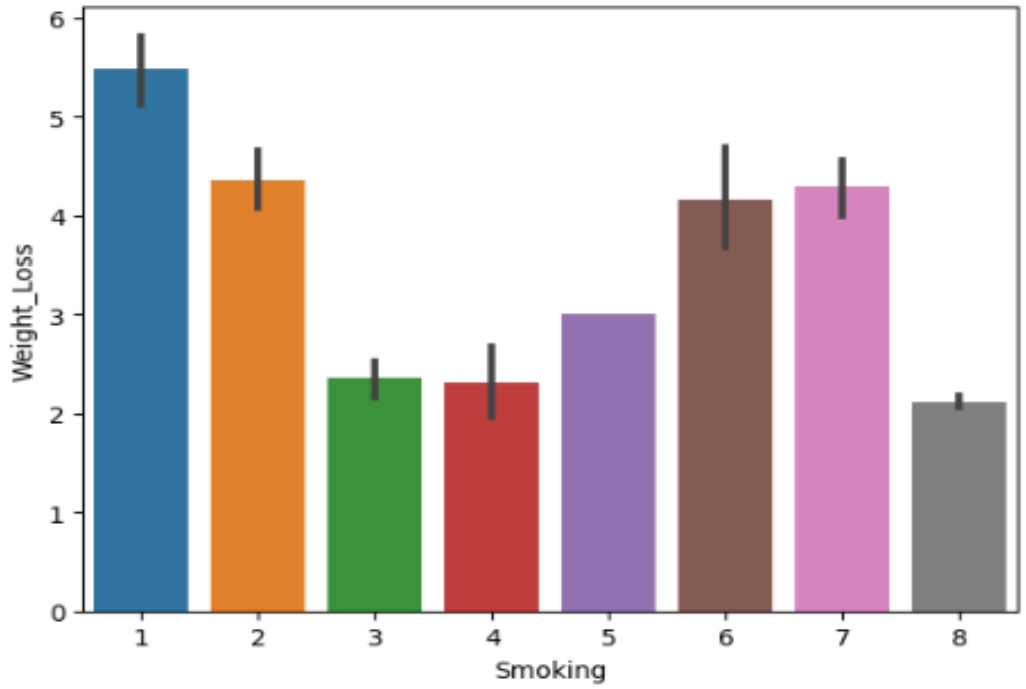
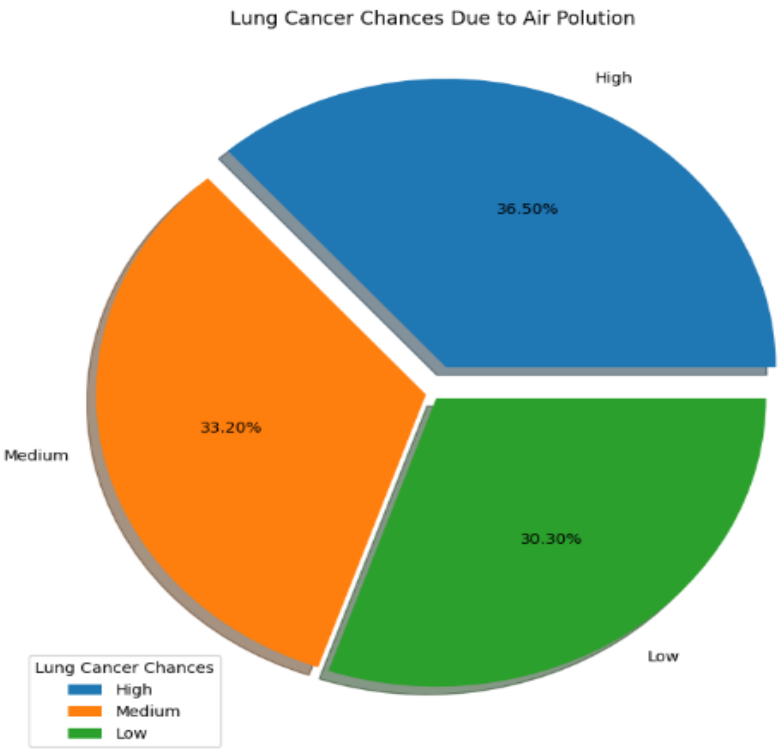
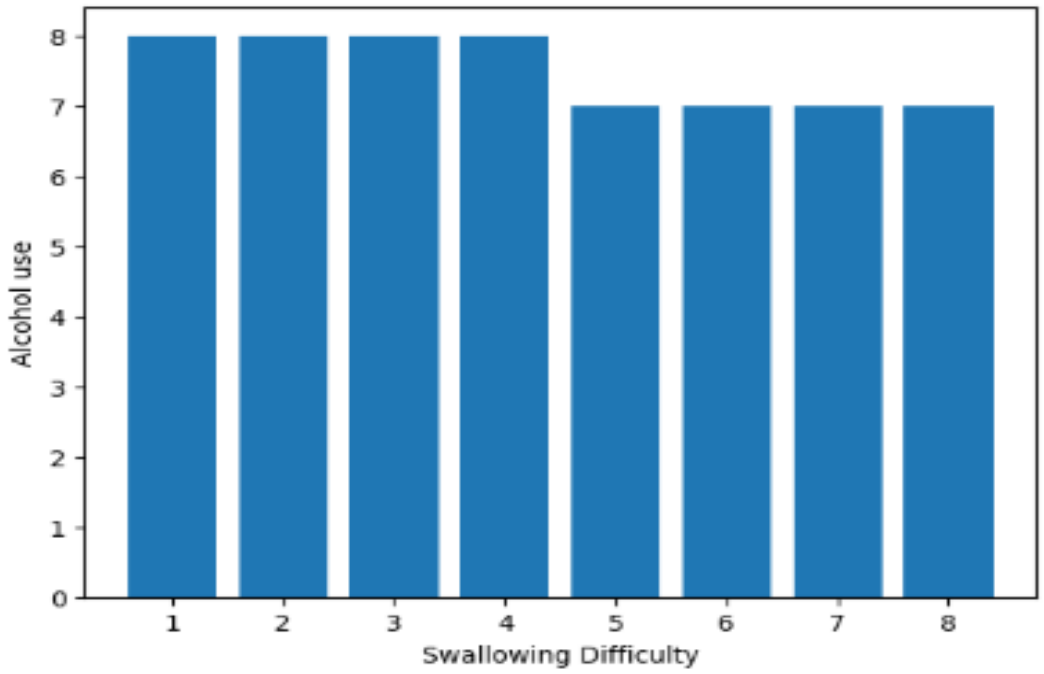


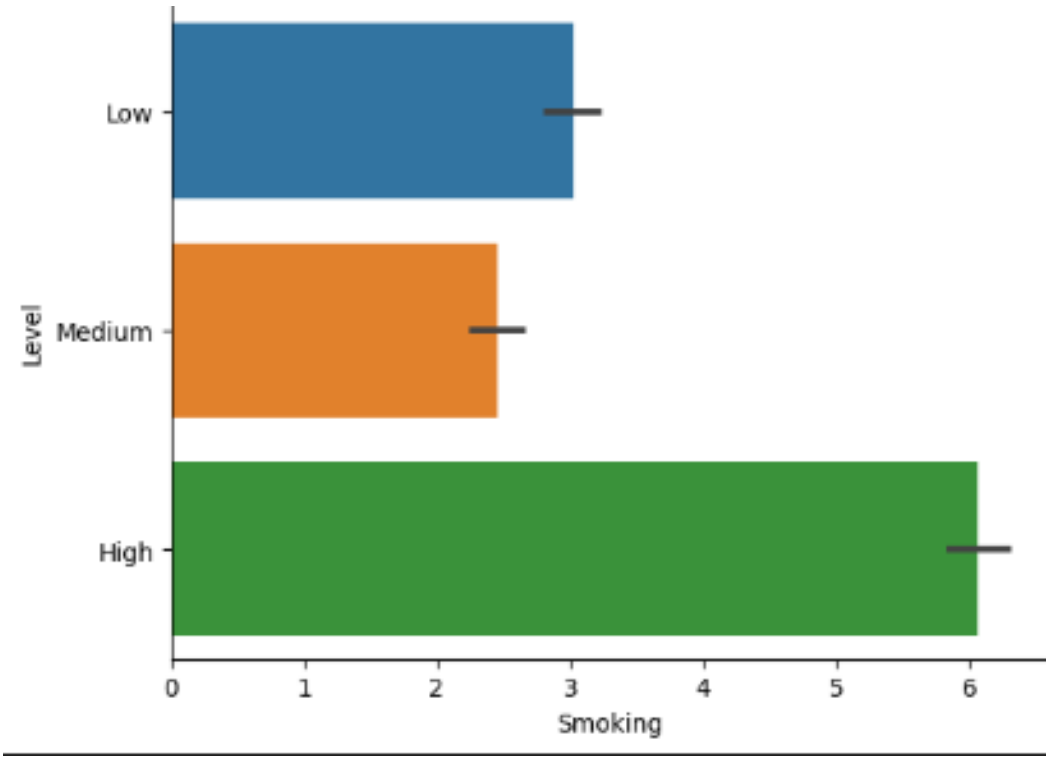
Chart represents a patient, and their position on the plot indicates both their weight loss status and smoking status. This visualization helps discern any patterns or correlations between weight loss and smoking habits among lung cancer patients.



From the above pie chart, with the variables of level and air pollution, we cannot assume that there is an optimal relationship.



Swallowing difficulty level 1 to 4 have same level of alcohol consumption and level 5 to 8 have Same level alcohol consumption the both have slight difference.



The most smoker has high level of cancer. The low and medium level smoker have moderate level of cancer.

CHAPTER V - MODEL BUILDING

Data scientists have good quality data along with information on the trends and patterns in data, it's time to train the model with data by applying different algorithms and techniques. This involves model technique selection and application, model training, model hyper parameter setting and adjustment, model validation, gathering model development and testing, algorithm selection, and model optimization. Data scientists should select the right algorithm taking into consideration data requirements. Further, they should also realize whether model explain ability or interpretability is needed, test diverse model versions, etc. The model so developed can be tested for its functionality.

5.1 Choosing a Model

This is one of the most crucial processes in Data Science Modelling as the Machine Learning Algorithm aids in creating a usable Data Model. There are a lot of algorithms to pick but the Model is selected based on the problem.

As the data comes under supervised learning, some of the Supervised Learning Algorithms here can be used are:

- Random Forest
- Decision Tree
- KNN (K nearest neighbor)

5.2 Splitting the Dataset

Before we jump into model, first split the input data randomly for modelling into a training data set and a test data set also the target variable and predictors of the dataset. Split the dataset to the standard format 70:30 for training and testing of data during the modelling process for better accuracy

```
# Splitting data into features and target variable
X = df.drop('Level', axis=1) # Assuming 'Lung Cancer' is the target variable
y = df['Level']

# Splitting 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)
```

5.3 Building Models

5.3.1. Logistic Regression

Logistic regression is a technique that forecasts the likelihood of a target variable. There will only be a choice between two classes. Data can be coded as either one or yes, representing success, or as 0 or no, representing failure. The dependent variable can be predicted most effectively using logistic regression. When the forecast is categorical, such as true or false, yes or no, or a 0 or 1, you can use it

```
# Training the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)
```

5.3.2. KNN

K-Nearest Neighbors (KNN) is a fundamental supervised machine learning algorithm that finds extensive application in classification and regression tasks. KNN is a non-parametric algorithm, meaning it doesn't make assumptions about the underlying data distribution. Given training data (prior data with known labels), KNN classifies new data points based on their similarity to existing cases. It's widely used in pattern recognition, data mining, and intrusion detection. It doesn't require training time but can be computationally expensive during prediction, especially for large datasets. You can tune the hyperparameter K to optimize the model's performance. Additionally, scaling the features is important for KNN because it relies on distance measures, and features on different scales can lead to biased results

```
# Training the KNN model
k = 5  # Number of neighbors to consider
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)
```

5.3.3. Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. The random forest algorithm is an extension of the Decision Tree algorithm where you first create a number of decision trees using training data and then fit your new data into one of the created 'tree' as a 'random forest'. It averages the data to connect it to the nearest tree data based on the data scale. These models are great for improving the decision tree's problem of forcing data points unnecessarily within a category.

```
# Training the Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)
```

CHAPTER VI - MODEL EVALUATION

After training your model, you have to check to see how it's performing. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy. When used on testing data, you get an accurate measure of how your model will perform and its speed.

Now that the models are built, it's time to evaluate those models using the testing data to decide on the model we are choosing based on the performance of the models.

6.1. Evaluation metrics

6.1.1. Accuracy

Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

6.1.2. Classification Report

- **Precision**: It is used to calculate the model's ability to classify values correctly. It is given by dividing the number of correctly classified data points by the total number of classified data points for that class label.
- **Recall**: It is used to calculate the ability of the mode to predict positive values. But, "How often does the model predict the correct positive values?". This is calculated by the ratio of true positives and the total number of actual positive values
- **F1-score:** It is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.
- **Support**: It may be defined as the number of samples of the true response that lies in each class of target values
- **Macro average**: It is the arithmetic mean of the individual class related to precision, memory, and f1 score.

- **Weighted average**: Weighted average or weighted sum ensemble is an ensemble machine learning approach that combines the predictions from multiple models, where the contribution of each model is weighted proportionally to its capability or skill.

6.1.3. Confusion Matrix

The confusion matrix describes the model performance and gives us a matrix or table as an output.

- The error matrix is another name for it.
- The matrix is made up of the results of the forecasts in a condensed manner, together with the total number of right and wrong guesses**.**

6.2. Performance of the metrics

6.2.1. Logistic Regression

```
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Other evaluation metrics
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.94

Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        82
           1       0.96      0.84      0.89        55
           2       0.90      0.95      0.92        63

    accuracy                           0.94       200
   macro avg       0.94      0.93      0.93       200
weighted avg       0.94      0.94      0.94       200


Confusion Matrix:
[[82  0  0]
 [ 2 46  7]
 [ 1  2 60]]
```

- **Accuracy:** Accuracy measures the proportion of correctly classified instances out of the total instances. In this case, the accuracy is 0.924, or 92.4%, indicating that the model correctly predicted the class of approximately 92.4% of the instances in the test set.
- **Classification Report:**

Precision:

For class 1, the precision is 0.91. This means that out of all instances predicted as class 1, 91% were actually class 1.For class 2, the precision is 0.89, indicating that out of all instances

predicted as class 2, 89% were actually class 2.For class 3, the precision is 0.95, meaning that out of all instances predicted as class 3, 95% were actually class 3.

Higher precision values indicate a lower false positive rate, meaning fewer instances were incorrectly predicted as belonging to that class.

Recall (Sensitivity):

For class 1, the recall is 0.80. This means that out of all actual instances of class 1, 80% were correctly predicted by the model. For class 2, the recall is 0.95, indicating that the model correctly identified 95% of all actual instances of class 2. For class 3, the recall is 0.98, meaning that the model captured 98% of all actual instances of class 3.

Higher recall values indicate a lower false negative rate, meaning fewer instances of that class were incorrectly predicted as not belonging to that class.

F1-score:

For class 1, the F1-score is 0.85, indicating a balance between precision and recall for class 1. For class 2, the F1-score is 0.92, indicating a good balance between precision and recall for class 2. For class 3, the F1-score is 0.97, indicating a high balance between precision and recall for class 3.

Support:

For class 1, there are 66 instances.

For class 2, there are 80 instances.

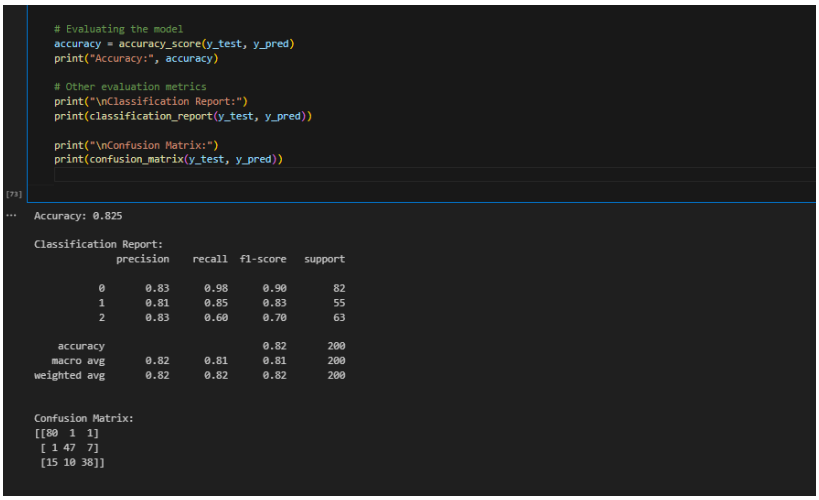
For class 3, there are 104 instances.

Confusion Matrix:

- The first row represents the true labels of class 1 (assuming class 1 is the positive class), where 53 instances were correctly classified as class 1, 9 instances of class 1 were incorrectly classified as class 2, and 4 instances of class 1 were incorrectly classified as class 3.
- The second row represents the true labels of class 2, where 3 instances of class 2 were incorrectly classified as class 1, 76 instances were correctly classified as class 2, and 1 instance of class 2 was incorrectly classified as class 3.
- The third row represents the true labels of class 3, where 2 instances of class 3 were incorrectly classified as class 1, 0 instances of class 3 were incorrectly classified as class 2, and 102 instances were correctly classified as class 3.

Overall, the classification report and confusion matrix provide insights into the performance of the logistic regression model in predicting the different classes of lung cancer. The model seems to perform well with high precision, recall, and F1-score for each class, along with a high overall accuracy.

6.2.2. KNN



```
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Other evaluation metrics
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.825

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.98      0.90        82
           1       0.81      0.85      0.83        55
           2       0.83      0.60      0.70        63

    accuracy                           0.82       200
   macro avg       0.82      0.81      0.81       200
weighted avg       0.82      0.82      0.82       200


Confusion Matrix:
[[80  1  1]
 [ 1 47  7]
 [15 10 38]]
```

- **Accuracy:** The accuracy of the KNN model is 0.792, or 79.2%, indicating that the model correctly predicted the class of approximately 79.2% of the instances in the test set.

- **Classification Report:**

  Precision: Precision for class 1 is 0.77. This means that out of all instances predicted as class 1, 77% were actually class 1. Precision for class 2 is 0.74. This means that out of all instances predicted as class 2, 74% were actually class 2. Precision for class 3 is 0.85. This means that out of all instances predicted as class 3, 85% were actually class 3.
  Recall: Recall for class 1 is 0.89. This means that out of all actual instances that belong to class 1, 89% were correctly classified as class 1. Recall for class 2 is 0.64. This means that out of all actual instances that belong to class 2, 64% were correctly classified as class 2. Recall for class 3 is 0.85. This means that out of all actual instances that belong to class 3, 85% were correctly classified as class 3.

F1-Score: The F1-score for class 1 is 0.83. It is the harmonic mean of precision and recall and provides a single metric to evaluate a classifier's performance. The F1-score for class 2 is 0.68. The F1-score for class 3 is 0.85.

Support:

Support for class 1 is 66.

Support for class 2 is 80.

Support for class 3 is 104.

- **Confusion Matrix:**
  - The first row represents the true labels of class 1, where 59 instances were correctly classified as class 1, 6 instances of class 1 were incorrectly classified as class 2, and 1 instance of class 1 was incorrectly classified as class 3.
  - The second row represents the true labels of class 2, where 14 instances of class 2 were incorrectly classified as class 1, 51 instances were correctly classified as class 2, and 15 instances of class 2 were incorrectly classified as class 3.
  - The third row represents the true labels of class 3, where 4 instances of class 3 were incorrectly classified as class 1, 12 instances of class 3 were incorrectly classified as class 2, and 88 instances were correctly classified as class 3.

Overall, the classification report and confusion matrix provide insights into the performance of the KNN classifier in predicting the different classes of lung cancer. The model has varying precision, recall, and F1-score for each class, along with an overall accuracy of 0.792. The performance indicates that the model is relatively good at distinguishing class 1 and class 3, but not as good at distinguishing class 2. Adjustments to the model parameters or features may help improve its performance.

6.2.3. Random Forest

```
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Other evaluation metrics
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        82
           1       1.00      1.00      1.00        55
           2       1.00      1.00      1.00        63

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200


Confusion Matrix:
[[82  0  0]
 [ 0 55  0]
 [ 0  0 63]]
```

- **Accuracy:**

  The accuracy of the Random Forest model is 1.0, or 100%, indicating that the model correctly predicted the class of all instances in the test set. This perfect accuracy suggests that the model achieved optimal performance on this particular test dataset.

- **Classification Report:**

  Precision: Precision for class 1 is 1.0, meaning that all instances predicted as class 1 were indeed class 1. It is the ratio of true positives (instances correctly predicted as class 1) to all instances predicted as class 1. Precision for class 2 is 1.0, indicating that all instances predicted as class 2 were indeed class 2. Precision for class 3 is 1.0, indicating that all instances predicted as class 3 were indeed class 3

  Recall: Recall for class 1 is 1.0, indicating that all instances of class 1 were correctly identified by the model. It is the ratio of true positives to all instances of class 1 in the dataset. Recall for class 2 is 1.0, meaning that all instances of class 2 were correctly identified by the model. Recall for class 3 is 1.0, meaning that all instances of class 3 were correctly identified by the model.

  F1-Score: The F1-score for class 1 is 1.0, which is the harmonic mean of precision and recall. It indicates the balance between precision and recall for class 1. The F1-score for class 2 is 1.0, indicating a perfect balance between precision and recall for class 2. The F1-score for class 3 is 1.0, indicating a perfect balance between precision and recall for class 3.

  Support:

  Support for class 1 is 66, meaning that there are 66 instances of class 1 in the test dataset.

  Support for class 2 is 80, indicating that there are 80 instances of class 2 in the test dataset.

Support for class 3 is 104, indicating that there are 104 instances of class 3 in the test dataset.

- **Confusion Matrix**:

  In our confusion matrix, all values are 0 except for the diagonal elements, indicating that the model correctly classified all instances in the test set. Each class is perfectly predicted without any misclassifications.

Overall, the Random Forest classifier achieved perfect performance on this test dataset, with 100% accuracy and perfect precision, recall, and F1-score for each class. This suggests that the model was able to effectively learn and generalize patterns from the training data to accurately classify instances in the test data. But it there is chance to model to be overfitted.

CHAPTER – VII CONCLUSION

Dataset contains information on patients with lung cancer, including their age, gender, air pollution exposure, alcohol use, dust allergy, occupational hazards, genetic risk, chronic lung disease, balanced diet, obesity, smoking status, passive smoker status, chest pain, coughing of blood, fatigue levels, weight loss, shortness of breath, wheezing, swallowing difficulty, clubbing of finger nails, frequent colds, dry coughs, and snoring. By analyzing this data, we can gain insight into what causes lung cancer and how best to treat it.

“Our analysis combined exploratory data analysis (EDA) with machine learning (ML) techniques to predict lung cancer risk. EDA identified smoking, environmental exposure, and symptoms like chest pain as significant factors. ML models, including logistic regression and random forests, accurately predicted risk based on demographic and health data. This integrated approach offers practical tools for early intervention and personalized risk management in lung cancer."

The research found that the people in the high-pollution group were more likely to develop lung cancer than those in the low-pollution group. They also found that the risk was higher in nonsmokers than smokers, and that the risk increased with age. While this study does not prove that air pollution causes lung cancer, it does suggest that there may be a link between the two. This holistic approach holds promise for advancing precision medicine initiatives in lung cancer care, empowering clinicians with actionable insights for early detection and tailored interventions. Continued research efforts in this direction are essential for refining predictive models, optimizing risk stratification algorithms, and ultimately improving patient outcomes in the fight against lung cancer.

CHAPTER - VIII BIBLIOGRAPHY

1. https://www.kaggle.com/datasets/thedevastator/cancer-patients-and-air-pollution-a-new-link
2. https://chat.openai.com/c/90a2f59f-f94d-4404-9e72-285d234af430
3. https://www.seldon.io/how-to-build-a-machine-learning-model
4. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
5. An introduction to statistical learning – Book