UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
SEMESTER I, SESSION 2025/2026

GROUP PROJECT REPORT

TITLE: **LUNG CANCER PREDICTION**

SECB3203: PROGRAMMING FOR BIOINFORMATICS
SECTION 01

LECTURER:
DR. SEAH CHOON SEN

GROUP:
02

NAME	MATRIC NUMBER
PAT YOON XIN	A24CS0292
EDWARD THIAN JIA MING	A24CS0244
ATIQAH QAISARA BINTI HIJAT	A24CS0228

ACKNOWLEDGEMENT

While authoring this project report, we had the opportunity to interact with a wide range of people who helped us understand and develop our views. We would like to extend our deepest gratitude to Dr Seah Choon Sen, our lecturer for Programming in Bioinformatics. His consistent support, direction, and critical input were invaluable in creating the final presentation of this report.

We would also like to thank our classmates for their significant assistance during this journey. In addition, we want to express our heartfelt appreciation to our families and those who have provided support on several occasions. Their perspectives and recommendations have been beneficial. Unfortunately, it is not feasible to mention everyone individually within the confines of this limited space, but we are deeply appreciative of the contributions made by others.

ABSTRACT

Accurate lung cancer risk prediction requires effective analysis of complex patient data that includes environmental, lifestyle, and clinical factors. This study uses a structured dataset of 1,000 patient records associated with lung cancer risk to develop a machine learning-based prediction framework. The methodology begins with data cleaning and preprocessing, including handling missing values and encoding categorical variables. The dataset is then divided into training and testing sets to ensure unbiased model evaluation. To identify both linear and non-linear relationships in the data, a number of predictive models are used, such as Multinomial Logistic Regression, Random Forest, and Gradient Boosting. The F1-score and confusion matrix are used to evaluate the model's performance, enabling a fair assessment of classification accuracy across lung cancer risk classes. The results demonstrate that ensemble-based models, particularly Random Forest and Gradient Boosting, provide improved predictive performance compared to traditional regression approaches.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Problem Background

Tobacco smoke and radon gas are major lung cancer risks because they produce reactive oxygen species (ROS) that damage DNA and cause mutations. While heavily damaged cells typically undergo apoptosis, ROS also influence cell growth and survival, strongly linking them to lung cancer development. Research shows that DNA repair levels affect how lung cells respond to this damage: low repair activity increases treatment sensitivity, while high repair activity contributes to drug resistance. However, despite extensive research on ROS and DNA repair, few lung cancer prediction models incorporate these biological factors. Most models rely primarily on basic risk factors like age, smoking habits, or medical history. This study addresses this research gap by exploring how ROS-related damage and DNA repair mechanisms can improve lung cancer prediction accuracy.

1.2 Problem Statement

Accurately predicting lung cancer risk and detecting the disease early requires analyzing large patient datasets with advanced computational methods. When risk assessment is inaccurate or incomplete, screening decisions may be less effective for individuals who need them most. Examining data on environmental exposures can uncover meaningful patterns that improve prediction. These data-driven insights support the creation of strong risk models that identify high-risk individuals who would benefit from targeted screening, enabling earlier detection when treatment is more effective, and survival rates are higher.

1.3 Aim

This project aims to develop a computational framework that integrates environmental, lifestyle, and clinical factors from lung cancer patient data to model biologically inspired functional devices for improving lung cancer risk prediction.

1.4 Objectives

The objectives of this project are shown as follows:

1. To gain a comprehensive understanding of lung cancer disease through a literature review and to provide a highly accurate prediction model.
2. To implement and develop a prediction model by applying deep learning for the early detection of lung cancer, allowing for timely and appropriate planning for treatment.
3. To test the effectiveness of the proposed prediction model in terms of accuracy and reliability.

1.5 Scopes

1.5.1 Domain of the Data

The dataset: https://www.kaggle.com/datasets/thedevastator/cancer-patients-and-air-pollution-a-new-link/data

The dataset contains 1,000 patient records, with each row representing a patient and 26 columns covering demographics, lifestyle, environment, medical history, and symptoms. Key factors include age, gender, air pollution exposure, allergies, occupational risks, smoking habits, alcohol use, diet, obesity, genetics, chronic lung disease, and clinical symptoms. The target variable shows each patient's lung cancer risk. The data's time frame and location are not specified, so findings only apply to this dataset.

1.5.2 Techniques To Be Used

The data will be cleaned and preprocessed, including encoding categories and handling missing values. If lung cancer risk classes are unbalanced, techniques like oversampling or undersampling will be applied. Several machine learning algorithms will be implemented, namely Multinomial Logistic Regression, Random Forest, and Gradient Boosting, to compare linear and ensemble-based prediction approaches. Model evaluation will

focus on the F1-score and confusion matrix, providing a balanced assessment of classification performance across different lung cancer risk categories.

1.5.3 Methodology

The study involves loading, cleaning, and preprocessing the dataset, then splitting it into training and testing sets. RFE will rank features by importance, and only the most predictive ones will be used in modelling. An exploratory analysis will examine how air pollution and lifestyle factors relate to lung cancer risk.

1.5.4 Limitations of The Research

The research is limited to the variables in the dataset and lacks clinical validation or longitudinal tracking. Time periods and geographic areas are unknown, so results cannot be generalized. Environmental data, like exact pollutant levels, are not included, and it is assumed that the dataset provider has managed ethical considerations for patient data.

CHAPTER 2: RESEARCH METHODOLOGY

2.1 Flowchart of The Proposed Approach

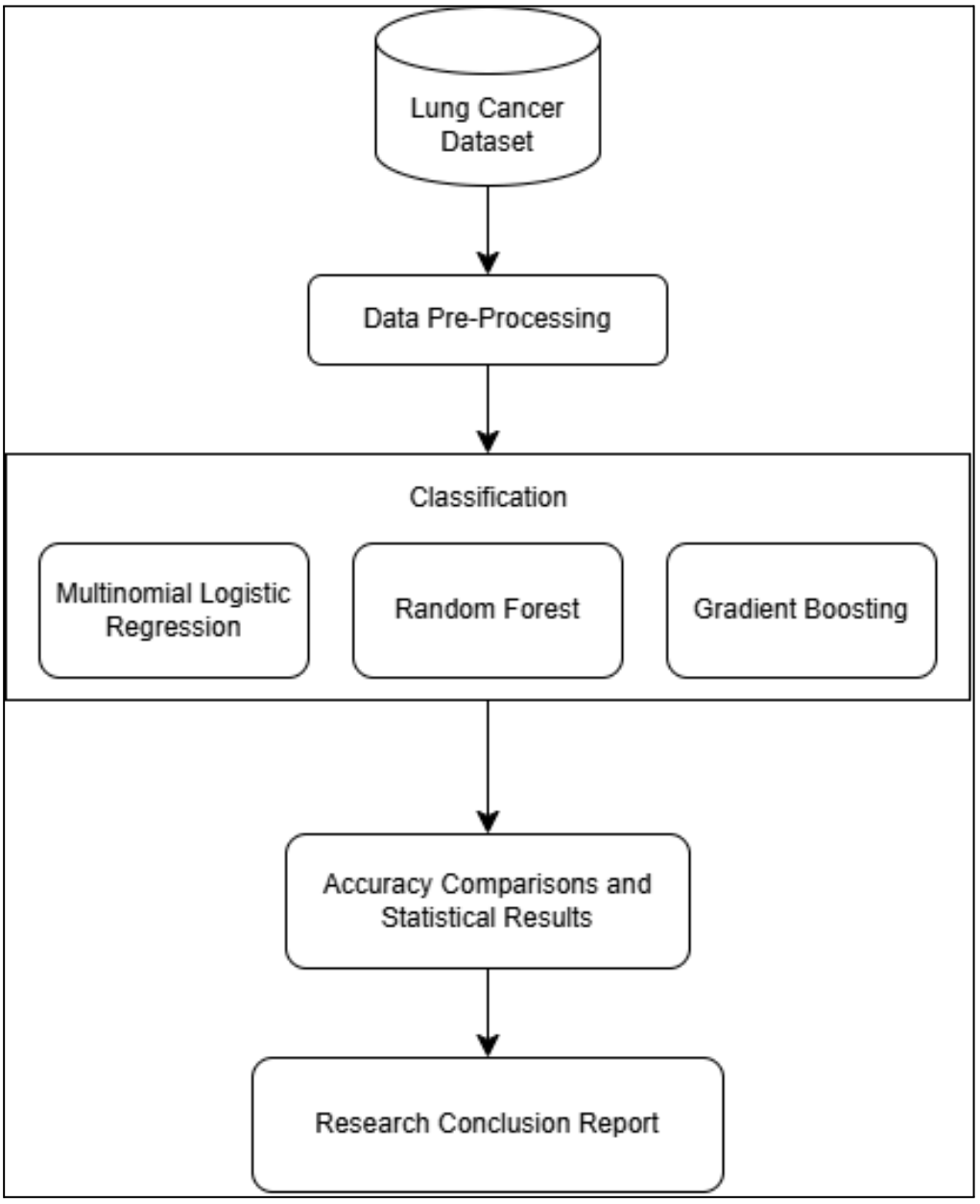

Figure 2.1: Flowchart of the Lung Cancer Prediction

2.2 Software and Hardware Requirements

2.2.1 Software Tools

1. Jupyter Notebook
    a. To implement code and perform data analysis, making it easier to demonstrate each step clearly in detail among team members.
    b. Allows the coding file (.ipynb) to be organized in a step-by-step manner.
    c. Uses Python as the centralized programming language of the project.
2. GitHub
    a. Used for version control and collaboration among team members and the lecturer.
    b. Enables code sharing, tracking changes, and monitoring project documentation.

2.2.2 Hardware Specifications

Hardware	Description
Processor - Intel Core i7 11800H	To speed up data processing and analysis
RAM - 8 GB	Sufficient storage in order handle large and complex datasets
System Type - 64-bit operating system, x64-based processor	Can support simulation, virtualization and process large data at once

CHAPTER 3: DATA PREPROCESSING

3.1 Understanding the Dataset

In this research, the dataset used includes patient data that may be related to the risk of lung cancer. It encompasses a variety of factors, including age, gender, and other lifestyle factors linked to lung cancer. These factors can be used as predictors to find trends and connections related to the incidence of lung cancer.

3.2 Importing and Exporting the Data

The initial step involves obtaining and importing the dataset for the project's analysis. The dataset, named "cancer_patient_data_sets," is sourced from Kaggle under "Lung Cancer Prediction."

```
df = pd.read_csv("cancer_patient_data_sets.csv", index_col='index')
display(df)
```

Figure 3.1: Code snippet indicating importing the dataset and displaying.

3.3 Getting Started Analyzing Data in Python

```
print(df.columns)
```

```
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')
```

Figure 3.2: Code snippet showing the column names of the dataset.

The above figure shows a code which displays all the column names in the cancer patient dataset. It shows an Index object containing 24 different columns.

The last line of the output (dtype = ‘object’) signifies that these column names are stored as text strings.

```
    print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, 0 to 999
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Patient Id              1000 non-null   object
 1   Age                     1000 non-null   int64
 2   Gender                  1000 non-null   int64
 3   Air Pollution           1000 non-null   int64
 4   Alcohol use             1000 non-null   int64
 5   Dust Allergy            1000 non-null   int64
 6   OccuPational Hazards    1000 non-null   int64
 7   Genetic Risk            1000 non-null   int64
 8   chronic Lung Disease    1000 non-null   int64
 9   Balanced Diet           1000 non-null   int64
 10  Obesity                 1000 non-null   int64
 11  Smoking                 1000 non-null   int64
 12  Passive Smoker          1000 non-null   int64
 13  Chest Pain              1000 non-null   int64
 14  Coughing of Blood       1000 non-null   int64
 15  Fatigue                 1000 non-null   int64
 16  Weight Loss             1000 non-null   int64
 17  Shortness of Breath     1000 non-null   int64
 18  Wheezing                1000 non-null   int64
 19  Swallowing Difficulty   1000 non-null   int64
...
 24  Level                   1000 non-null   object
dtypes: int64(23), object(2)
memory usage: 203.1+ KB
None
```

Figure 3.3: Code snippet resulting in the DataFrame’s summary.

Next, by using the code based on the figure above, it is a method in Pandas that allows for providing a summary of the DataFrame by quickly assessing its structure as well as optimising memory usage.

This fragment code includes the number of rows in the DataFrame, column names and their associated data types. It can be seen that the dataset has 1000 entries with index values ranging from 0 to 999, and a total of 25 columns. All columns contain 1000 non-null values, indicating that there are no missing values in the dataset.

```
df.drop(columns=['Patient Id'], inplace=True)
print("Duplicate rows before:", df.duplicated().sum())

df.drop_duplicates(inplace=True)

print("Duplicate rows after:", df.duplicated().sum())
print(df.isna().sum())
```

```
Duplicate rows before: 848
Duplicate rows after: 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
```

Figure 3.4: Code snippet of dropping ID and duplicated rows in the dataset.

Furthermore, the use of the drop() function has been made to remove the Patient Id column from the Data Frame. This attribute function is solely focused on a unique identifier for each record and does not provide predictive value for the learning process. Therefore, it is excluded to prevent unnecessary complexity and potential bias during model training.

Additionally, there is also a method to identify duplicated samples in the dataset. It can be seen that there are 848 samples from the 1000 samples that are duplicated. After using the drop_duplicates() function, it is confirmed that there are no duplicates in the dataset.

```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
print('Cancer Levels: ', df['Level'].unique())
map = {'High': 2, 'Medium': 1, 'Low': 0}
df["Level"].replace(map, inplace=True)
print('Cancer Levels: ', df['Level'].unique())
```

```
Cancer Levels:  [0 1 2]
Cancer Levels:  [0 1 2]
```

Figure 3.5: Code snippet for encoding the cancer level into numerical values.

Then, the figure above shows the encoding process applied to the categorical target variable, Cancer Level. Warning messages have been added to enhance the readability of the output, by using a mapping technique to convert the categorical labels: Low, Medium, High into numerical values of 0, 1, and 2, respectively.

The unique() function is used both before and after the transformation, ensuring that the encoding procedure has been completed, resulting in its representation as a numerical format suitable for use by machine learning algorithms.

CHAPTER 4: EXPLORATORY DATA ANALYSIS

4.1. Univariate Analysis

Univariate analysis is the simplest form of data analysis, which focuses on only one variable at a time to describe its characteristics. Here, the cancer level "Level" variable is used in conducting the univariate analysis to help understand the cancer level. Firstly, the dataset was separated into features (X) and the target variable (y) for clarity in analysis. The following data preparation was carried out to double confirm that there are no missing values in the target variable and to ensure complete data for all observations.

```
X = df.drop(columns='Level')
y = df.Level

display(X.head())
print(y[:5])
```

```
index
0    0
1    1
2    2
3    2
4    2
Name: Level, dtype: int64
```

Figure 4.1: Data Preparation of target variable "Level"

Following on is a pie chart to visualise the distribution of the three categorical levels (High, Medium, and Low) of the cancer level. It is shown that there are 53 observations (34.87%) in the high cancer level, while 52 observations (34.21%) are in the medium level, and 47 observations (30.92%) in the low level. This indicates that there is no significant bias in the dataset and it provides adequate samples for training models to recognise patterns across all categories correctly.

```
plt.figure(figsize=(10,6))
plt.pie(df['Level'].value_counts(),
        labels=map.keys(),
        autopct=lambda p: '{:.2f}%\n{:,.0f}'.format(p, p * sum(df['Level'].value_counts()) / 100))
plt.title('Distribution of Levels')
plt.show()
```

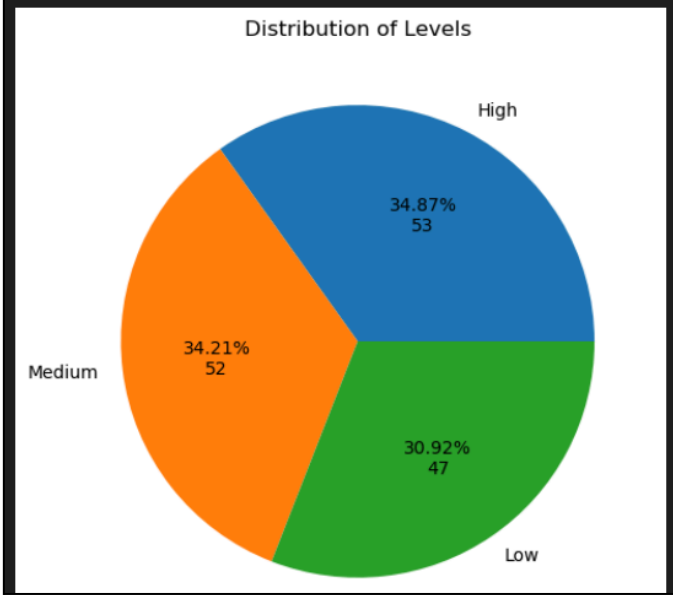

Figure 4.2: Pie Chart of target variable "Level"

4.2. Correlation Matrix

The correlation matrix shown below helps identify relationships between all variables in the dataset. The darkest blue indicates the strongest positive correlation, while lighter colours show weaker or negative correlation strength.

```
plt.figure(figsize=(20,15))
sns.heatmap(df.corr(), annot=True, cmap=plt.cm.PuBu)
plt.show()
```

Figure 4.3: Code of Correlation Matrix

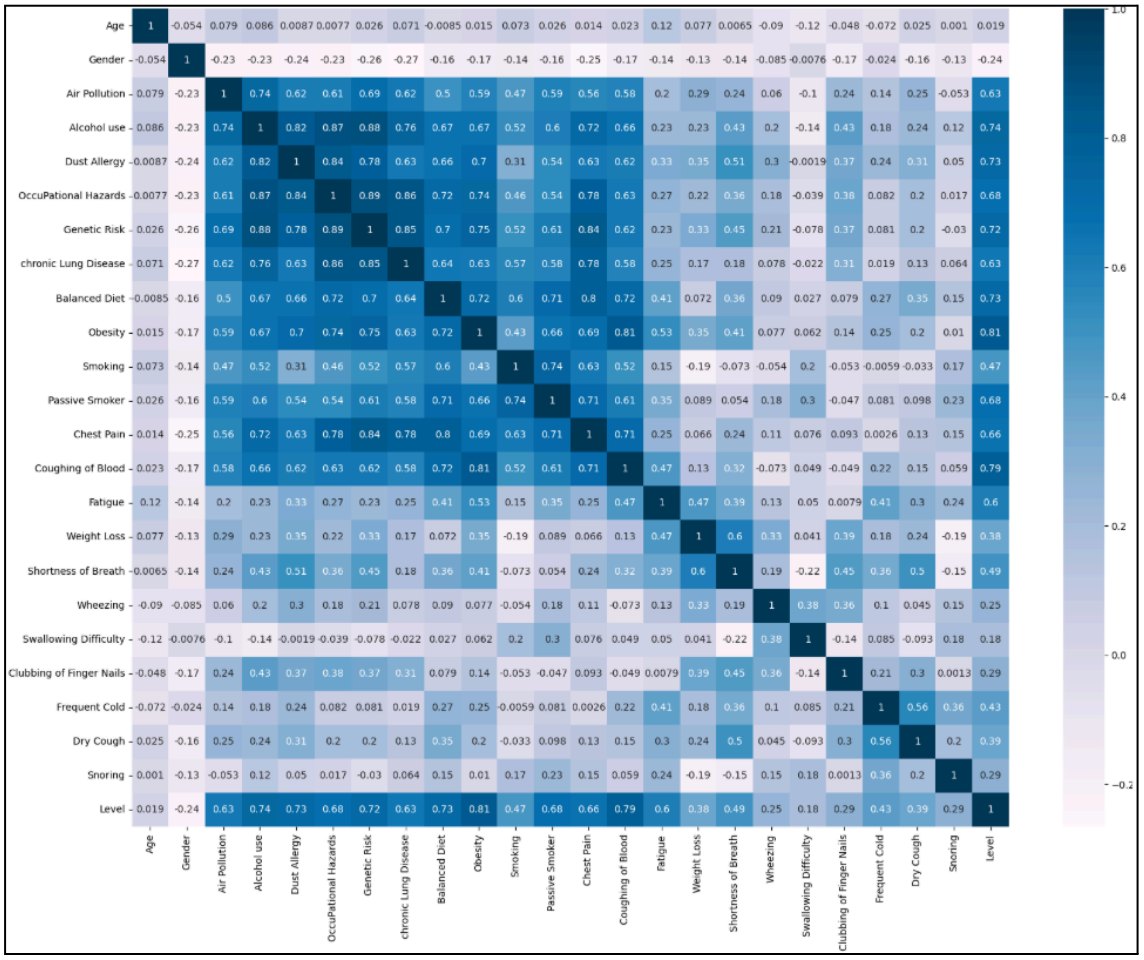

Figure 4.4: Correlation Heatmap

Based on Figure 4.4, various pairs of variables have demonstrated strong positive correlations with r > 0.7. Some of the most noticeable pairs is listed as below:

1. **Chronic Lung Disease and Occupational Hazards (r = 0.86)**

   This is a strong causal relationship. It is proven that a workplace where prolonged exposure to dust or other chemical substances significantly increases the risk of chronic lung diseases such as silicosis. There are evidence for some hazards where exposures in childhood or before birth can potentiate future non-malignant respiratory disease. *(Pacyna & Pacyna, 2016)*.

2. **Chronic Lung Disease and Genetic Risk (r = 0.85)**

   This is also a strong causal relationship. Family history and genetic markers such as alpha-1 antitrypsin deficiency can significantly increase the risk of chronic lung diseases.

3. **Lung Cancer Level and Obesity (r = 0.81)**

Obesity is surprisingly high associated with lung cancer. Obesity may be associated with lifestyle factors that worsen cancer severity.

4. **Genetic Risk and Chest Pain (r = 0.84)**

   This indicates that people with genetic risk factors have a high possibility to develop respiratory problems that have a symptom as chest pain.

5. **Chronic Lung Disease and Chest Pain (r = 0.78)**

   This indicates that chest pain can serve as an accurate indicator of chronic lung diseases.

Overall, this correlation analysis proves that health conditions such as obesity and genetic risks are strongly associated with the chronic lung diseases and cancer risks. The strong correlations among the variables show that they are closely related and describe similar aspects of a person's respiratory health. This pattern aligns with domain knowledge about risk factors for respiratory conditions.

4.3. Bivariate Data Analysis

4.3.1. Scatter Plot

Scatter plots are plotted to visualise the relationships between individual predictor variables and the target variable 'Level' (lung cancer severity). The scatter plots are shown below:

```
fig, ax = plt.subplots(ncols=4, nrows=6, figsize=(20, 20))
ax = ax.flatten()

numeric_cols = df.select_dtypes(include='number').columns.drop('Level', errors='ignore')

for i, col in enumerate(numeric_cols):
    sns.regplot(
        x=col,
        y='Level',
        data=df,
        lowess=True,
        color=colors[i % len(colors)],
        ax=ax[i]
    )
    ax[i].set_title(col.title())
    ax[i].set_xlabel(col)
    ax[i].set_ylabel('Level')

plt.tight_layout(pad=0.5, w_pad=0.3, h_pad=1.0)
plt.show()
```

Figure 4.5: Code of Scatter Plots

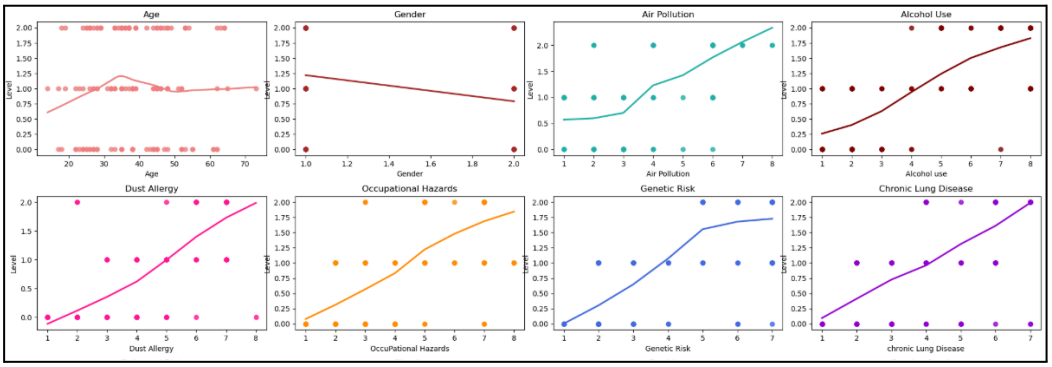

Figure 4.5: Scatter Plots (1)

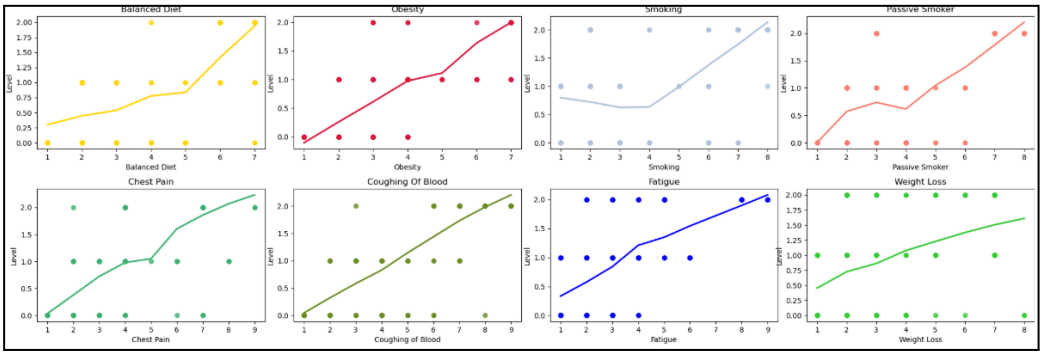

Figure 4.6: Scatter Plots (2)

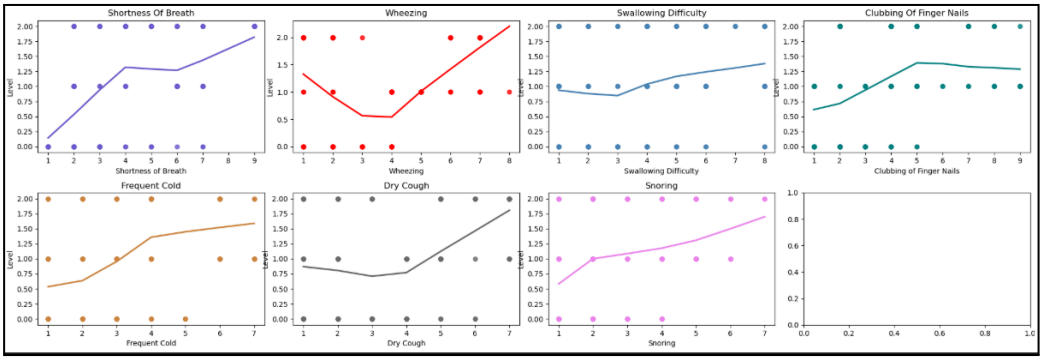

Figure 4.7: Scatter Plots (3)

Among the scatter plots, 3 most notable plots are worth mentioning. The first one is "Coughing of Blood vs. Level" which shows the strongest positive relationship in the dataset. The line rises steeply and data points show clear clustering: Level 0 (low cancer level) patients rarely experience coughing blood; while Level 2 (high cancel level) patients suffer from coughing blood severely. From the analysis, it can be interpreted that coughing blood is a critical diagnostic indicator of lung cancer and can be a key feature when predicting cancer severity.

Another notable plot is "Chronic Lung Disease vs. Level" which also shows a strong positive relationship. This can be interpreted that pre-existing

chronic lung disease is a major risk factor that can contribute to severe lung cancer.

Lastly, plot "Alcohol Use vs. Level" also shows a strong positive relationship. This indicates that people who consume alcohol have higher possibility to get lung cancer as alcohol is proven as a carcinogen, a type of cancer-causing substance (*Alcohol and Cancer Risk Fact Sheet*, 2025).

4.3.2. Violin Plot

Violin plots are also plotted to show the distribution shape of each feature grouped by cancer levels. The plots are as shown as below:

```
fig, ax = plt.subplots(ncols=4, nrows=6, figsize=(20, 20))
ax = ax.flatten()

for i, col in enumerate(df.select_dtypes(include='number').columns):
    sns.violinplot(
        x=df['Level'],
        y=df[col],
        palette='turbo',
        ax=ax[i]
    )
    ax[i].set_title(col.title())
    ax[i].set_xlabel('Level')
    ax[i].set_ylabel(col)

plt.tight_layout(pad=0.5, w_pad=0.2, h_pad=2.5)
plt.show()
```

Figure 4.8:Code of Violin Plots

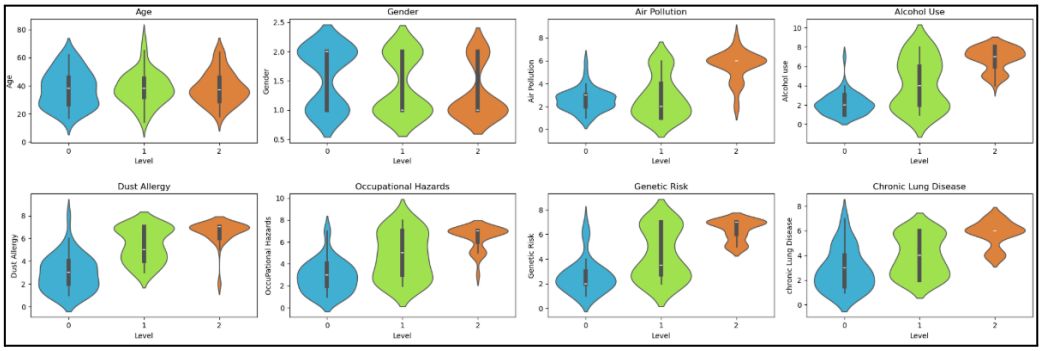

Figure 4.9: Violin Plots (1)

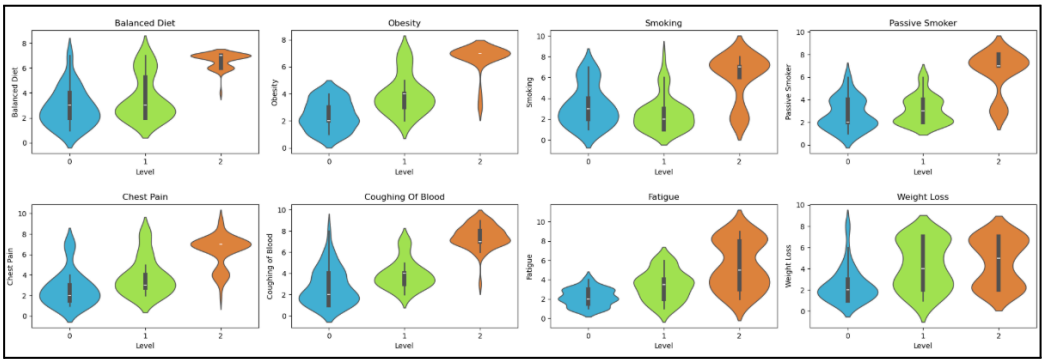

Figure 4.10: Violin Plots (2)

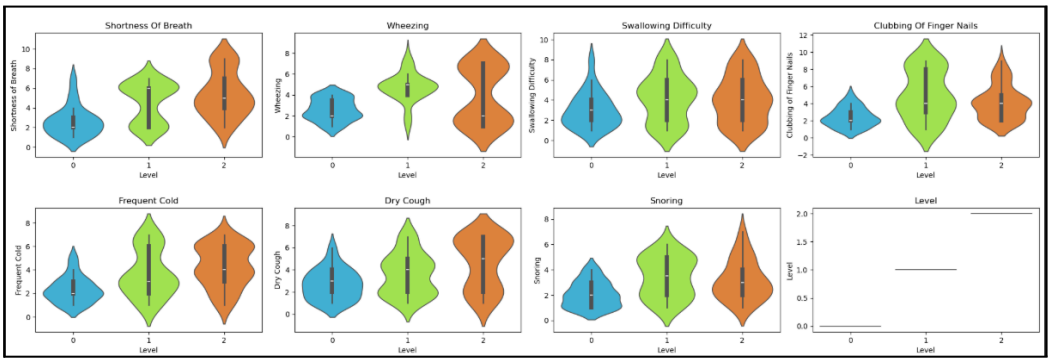

Figure 4.11: Violin Plots (3)

Among all the violin plots, the plot “Alcohol Use”, which shows the clearest separation, is worth mentioning. At level 0 (low cancel level), there is a narrow distribution concentrated at low values (0-4), with median around 1-2. While at level 1 (medium cancel level), there is a broader distribution spanning values 0-10, with median around 5-6. Whereas at level 2 (high cancer level), there is a concentration at high values (4-8), with median around 7-8. The consistent shift in distribution and minimal overlap between extreme levels suggests this variable has high predictive power in predicting cancer severity.

Overall, Alcohol Use, Air Pollution, Smoking, Passive Smoker, Fatigue, Shortness of Breath has good separation, while Age, Gender, Wheezing, Swallowing Difficulty, Dry Cough has poorer separation.

4.3.3. Histogram

Histograms are also plotted to show the shape of each variable's distribution. The histograms are as shown as below:

```
fig, ax = plt.subplots(ncols=8, nrows=3, figsize=(20, 10))
ax = ax.flatten()

i = 0
for col in df.select_dtypes(include='number').columns:
    sns.histplot(
        df[col],
        bins=20,
        color=colors[i],
        ax=ax[i]
    )
    ax[i].set_title(col.title())
    ax[i].set_xlabel(col)
    ax[i].set_ylabel('Frequency')
    i += 1

plt.tight_layout(pad=0.2, w_pad=0.2, h_pad=2.5)
plt.show()
```

Figure 4.12: Code of Histograms

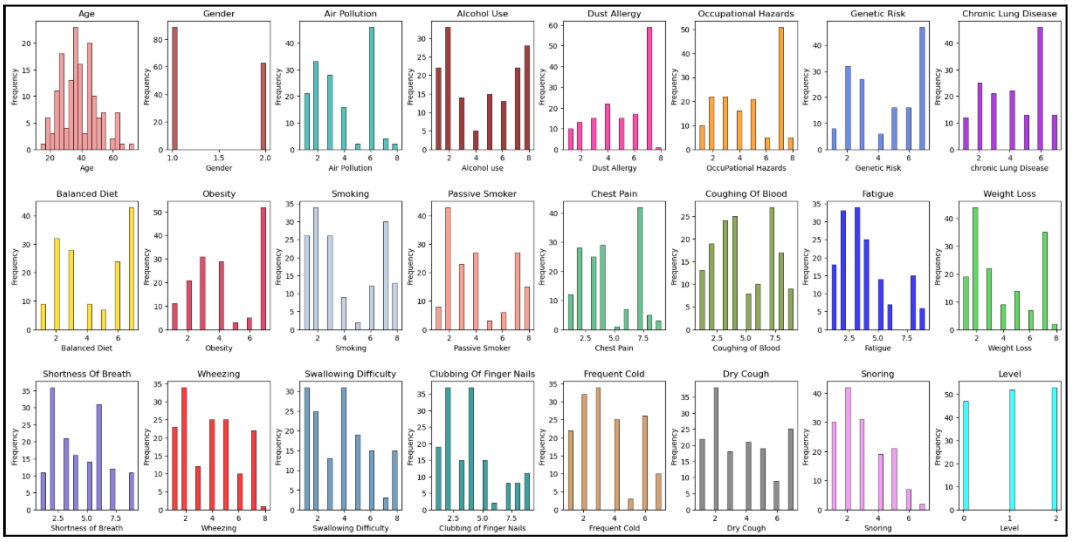

Figure 4.13:Histogram

Among all the histograms, the "Dust Allergy" which has a highly concentrated distribution, is worth mentioning. The histogram is right-skewed with a massive spike at value 7. This can be interpreted as the lung cancer patients generally have a high prevalence of severe dust allergies. The same interpretation applies to the variables "Occupational Hazards" and "Genetic Risk".

CHAPTER 5: MODEL DEVELOPMENT

5.1. Data Splitting for Model Training

In order to develop models to predict lung cancer cases, firstly it is important to split the data adequately. Here, the dataset is splitted into ratio 7:3, where 70% of data is used to train the model while 30% of data is used to test the model. X_train represents the variables (features) for training; x_test represents features for testing; Y_train represents cancer "Level" for training; and y_test represents cancer "Level" for testing. In the result, there are 106 observations (23 features each) for training and 46 observations for testing. The training output shows Level 2 (high cancer level) has 43 counts (40.6%), Level 1 (medium cancer level) has 30 counts (28.3%), and Level 0 (low cancer level) has 33 counts (31.1%). Overall, this shows a balanced distribution and there is no bias that will cause incorrect prediction. The data splitting code is as shown below:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

print(f'Shapes - X Training: {X_train.shape} and X Testing {X_test.shape}')
print(f'Shapes - Y Training: {y_train.shape} and Y Testing {y_test.shape}')

print(f'\nTraining output counts\n{y_train.value_counts()}')
```

```
Shapes - X Training: (106, 23) and X Testing (46, 23)
Shapes - Y Training: (106,) and Y Testing (46,)

Training output counts
Level
2    43
0    33
1    30
Name: count, dtype: int64
```

Figure 5.1:Data Splitting

5.2. Multinomial Logistic Regression

The first model that is implemented is Multinomial Logistic Regression. It is a classification algorithm that extends logistic regression to handle multiple classes. It models the probability of each class and predicts the class with the highest probability. The implementation of Multinomial Logistic Regression is as shown below.

```
MR = LogisticRegression(multi_class='multinomial', solver='newton-cg')

MR.fit(X_train, y_train)

MR_pred = MR.predict(X_test)

print('Intercept: \n', MR.intercept_)
print('Coefficients: \n', MR.coef_)
```

```
Intercept:
 [ 18.51399683   0.81421937 -19.3282162 ]
Coefficients:
 [[ 0.02208184  0.06938152 -0.22377607 -0.26743899 -0.39075575  0.05706996
  -0.2133126  -0.11452469 -0.08767215 -0.34435     0.22591694 -0.21341707
  -0.04847804 -0.61309231 -0.38722368 -0.2782542  -0.11285895 -0.4641843
  -0.30626139 -0.53071048 -0.41520455 -0.03606263 -0.26925021]
 [-0.01080677 -0.05581483 -0.22405537 -0.01346063  0.38077848 -0.12061795
   0.13570516 -0.00371873 -0.14843305  0.0572819  -0.51980575 -0.14573813
   0.07769307  0.18436355 -0.08802874 -0.06703969 -0.14697198  0.2902913
   0.16851362  0.46366918  0.14876101 -0.23153555  0.09411865]
 [-0.01127507 -0.01356669  0.44783144  0.28089962  0.00997728  0.06354799
   0.07760744  0.11824342  0.23610521  0.2870681   0.29388881  0.3591552
  -0.02921504  0.42872875  0.47525242  0.34529389  0.25983094  0.173893
   0.13774777  0.06704129  0.26644353  0.26759818  0.17513156]]
```

Figure 5.2: Implementation of Multinomial Logistic Regression

5.3. Random Forest Classifier

The second model that is conducted is Random Forest Classifier. It is an ensemble learning method that builds various decision trees during the training phase and merge the predictions together to produce a highly accurate and stable classification. The implementation of Random Forest Classifier is as shown below.

```
def random_forest_n_best(X_train, y_train, X_test, y_test, n_list):

    scores = []

    for n in n_list:
        RF = RandomForestClassifier(n_estimators=n)
        RF.fit(X_train, y_train)
        RF_pred = RF.predict(X_test)

        scores.append(accuracy_score(y_test, RF_pred))

    plt.plot(n_list, scores)
    plt.xlabel('Number of Estimators')
    plt.ylabel('Testing Accuracy')
    plt.title('Random Forest Accuracy vs n_estimators')
    plt.grid(alpha=0.1)
    plt.show()
```

Figure 5.3: Implementation of Random Forest Classifier

To optimize the Random Forest model, it is important to evaluate classification accuracy from the number of estimators (trees). Here, models with n_estimators ranging from 1 to 20 are trained and tested.

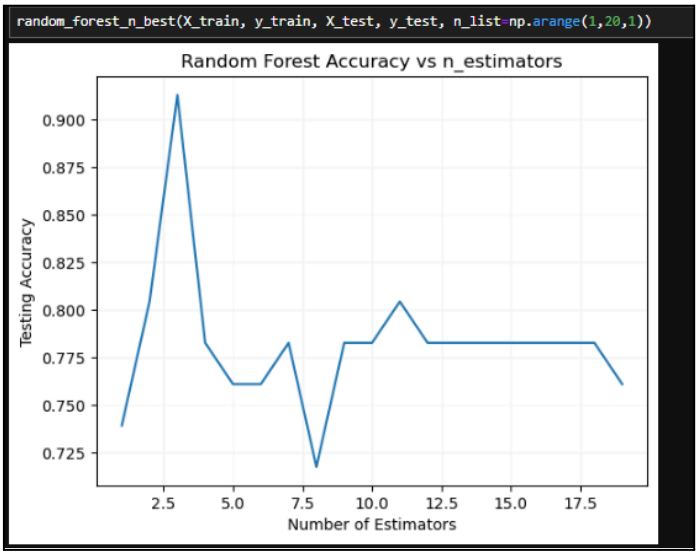

Figure 5.3: Random Forest Accuracy VS n_estimators

It is shown that the the accuracy starts at approximately 0.74 with 1 tree, then increases and reaches its highest value of about 0.91 at 3 trees. After this peak, the accuracy drops to around 0.78 at 4 trees and continues to fluctuate between approximately 0.72 and 0.80 as the number of trees increases. From around 10 to 18 trees, the accuracy stabilizes at roughly 0.78, before slightly decreasing again to about 0.76 at 19–20 trees. Overall, the graph indicates that the best testing accuracy is achieved with a small number of trees, while adding more trees beyond this point does not lead to further improvement.

Prediction analysis of Random Forest Classifier is also conducted as shown below. As a result, there are 14 predictions (30.4%) in Level 2 (high cancer level), 11 predictions (26.8%) in Level 1 (medium cancer level) and 21 predictions (51.2%) in Level 0 (low cancer level). Overall, it matches the actual distribution.

```
RF = RandomForestClassifier(n_estimators=3)

RF.fit(X_train, y_train)

RF_pred = RF.predict(X_test)
pd.Series(RF_pred).value_counts()
```

```
0    21
2    14
1    11
Name: count, dtype: int64
```

Figure 5.4: Prediction analysis of Random Forest Classifier

5.4. Gradient Boosting Classifier

The third model that is conducted is Gradient Boosting Classifier. It is a powerful ensemble learning method that combines various "weak" prediction models sequentially to produce a highly accurate classifier. The implementation of Gradient Boosting Classifier is as shown below.

```
def gradient_boosting_n_best(X_train, y_train, X_test, y_test, n_list):
    scores = []

    for n in n_list:
        GB = GradientBoostingClassifier(n_estimators=n)
        GB.fit(X_train, y_train)
        y_pred = GB.predict(X_test)
        scores.append(accuracy_score(y_test, y_pred))

    plt.plot(n_list, scores, marker='o')
    plt.xlabel('Number of Estimators')
    plt.ylabel('Testing Accuracy')
    plt.title('Gradient Boosting Accuracy vs n_estimators')
    plt.grid(alpha=0.2)
    plt.show()
```

Figure 5.5: Implementation of Gradient Boosting Classifier

Evaluation of classification accuracy from the number of estimators (trees) is also conducted as shown below. Here, models with n_estimators ranging from 1 to 20 are trained and tested.

```
n_list = np.arange(1, 20, 1)
gradient_boosting_n_best(X_train, y_train, X_test, y_test, n_list)
```

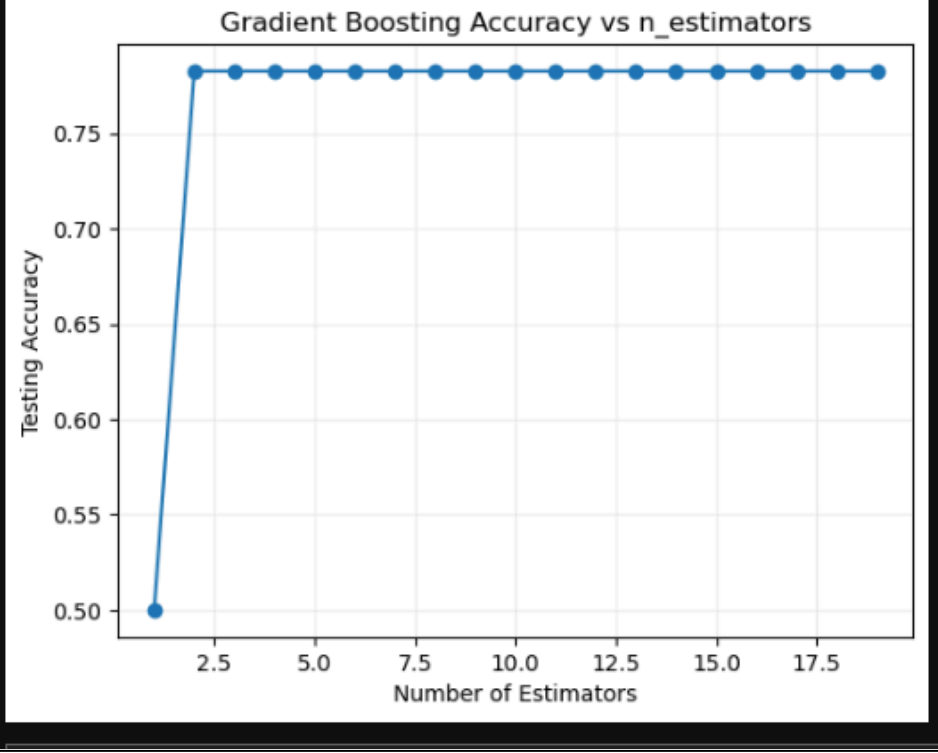

Figure 5.6: Gradient Boosting Accuracy VS n_estimators

It is shown that the accuracy of the model is below 1.00 during the first tree, at approximately 0.50. After adding more trees, the accuracy increases sharply to around 0.78 by the second tree and then remains stable at this level up to 20 trees. This demonstrates that the sequential error-correction mechanism of the Gradient Boosting Classifier is effective, as most of the performance improvement is achieved in the early stages of boosting, while additional trees provide minimal further gains.

Prediction analysis of the model is also conducted as shown below.

```
GB = GradientBoostingClassifier(n_estimators=100)

GB.fit(X_train, y_train)

GB_pred = GB.predict(X_test)
```

Figure 5.7: Prediction analysis of Gradient Boosting Classifier.

CHAPTER 6: MODEL EVALUATION

6.1. Model Evaluation Metrics

Evaluation Metrics such as Accuracy, Recall, F1-score and Precision are used to assess and compare the performance of the 3 different classification models. Firstly, accuracy is to measure the proportion of correctly predicted samples to the total number of samples. It is defined as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

where:

TP = True Positive;

TN = True Negative;

FP = False Positive;

FN = False Negative

Furthermore, precision is used to measure the proportion of correctly predicted positive samples among all samples predicted as positive. It is defined as:

$$Precision = \frac{TP}{TP+FP}$$

Recall is also used to measure the proportion of actual positive samples that are correctly identified by the model. It is defined as:

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

Lastly, F1-score is used to measure the overall precision and recall. It is defined as:

$$F1\ score = 2\times\frac{Precision\times Recall}{Precision+Recall}$$

6.2. Multinomial Logistic Regression

In order to evaluate the performance of the Multinomial Logistic Regression model in this dataset, confusion matrix analysis and classification report of the model are conducted as shown below. Their respective outputs are also shown below.

```python
def CM(y_test, y_pred_test, col_names, title='', cmap=plt.cm.Blues):

    CM = confusion_matrix(y_test, y_pred_test)
    plt.figure(figsize=(9,7))
    sns.heatmap(CM, annot=True, annot_kws={'size':15}, fmt=".0f", cmap=cmap, linewidths=5)

    tick_marks = np.arange(len(col_names))
    plt.xticks(tick_marks + 0.5, col_names)
    plt.yticks(tick_marks + 0.5, col_names, rotation=0)
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.title('Confusion Matrix ' + title)
    plt.show()
```

Figure 6.1: Implementation of Confusion Matrix Analysis of Multinomial Logistic Regression

```python
CM(y_test, MR_pred, col_names=['Low', 'Medium', 'High'], title='- Multinomial Regression')

ml_accuracies['Multinomial Model'] = accuracy_score(y_test, MR_pred)
print(classification_report(y_test, MR_pred))
```

Figure 6.2: Implementation of Classification Report of Multinomial Logistic Regression

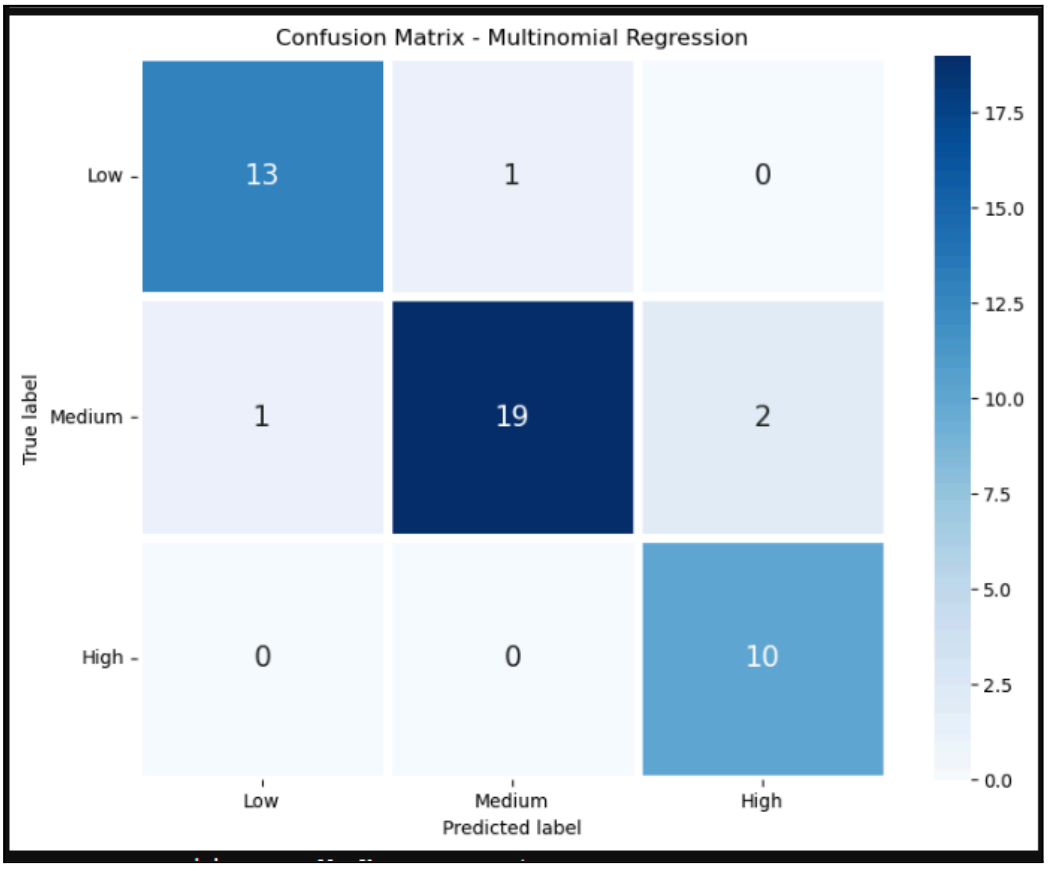

Figure 6.3: Confusion Matrix of Multinomial Logistic Regression

	precision	recall	f1-score	support
0	0.93	0.93	0.93	14
1	0.95	0.86	0.90	22
2	0.83	1.00	0.91	10
accuracy			0.91	46
macro avg	0.90	0.93	0.91	46
weighted avg	0.92	0.91	0.91	46

Figure 6.4: Classification Report of Multinomial Logistic Regression

According to the confusion matrix in Figure 6.3, the model demonstrates strong classification performance. Of the 14 low cases, 13 were correctly predicted with 1 misclassified as medium. For the 22 medium cases, 19 were correctly predicted, with 1 misclassified as low and 2 as high. All 10 high cases were correctly predicted with no misclassifications.

The classification report in Figure 6.4 shows that the model achieved high performance across all metrics. The precision ranges from 0.93 to 0.95, recall from 0.86 to 1.00, and F1-scores from 0.90 to 0.93 across the three classes. The overall accuracy of the model is 0.91 (91%), with macro average and weighted average scores also at 0.91, indicating balanced performance across all classes.

6.3. Random Forest Classifier

Confusion matrix analysis and classification report of the Random Forest Classifier model are also conducted as shown below. Their respective outputs are also shown below.

```
CM(y_test, RF_pred, col_names=['Low', 'Medium', 'High'], title='- Random Forest', cmap='Purples')

ml_accuracies['Random Forest'] = accuracy_score(y_test, RF_pred)
print(classification_report(y_test, RF_pred))
```

Figure 6.5: Implementation of Confusion Matrix and Classification Report of Random Forest Classifier

Figure 6.6: Confusion Matrix of Random Forest Classifier

	precision	recall	f1-score	support
0	0.62	0.93	0.74	14
1	0.91	0.45	0.61	22
2	0.71	1.00	0.83	10
accuracy			0.72	46
macro avg	0.75	0.79	0.73	46
weighted avg	0.78	0.72	0.70	46

Figure 6.7: Classification Report of Random Forest Classifier

According to the confusion matrix in Figure 6.6, the model demonstrates strong classification performance. Of the 14 low cases, 13 were correctly predicted, with 8 misclassified as medium. For the 22 medium cases, 10 were correctly predicted, with 8 misclassified as low and 4 as high. All 10 high cases were correctly predicted with no misclassifications.

The classification report in Figure 6.7 shows that the model achieved high performance across all metrics. The precision ranges from 0.62 to 0.93, recall from 0.45to 1.00, and F1-scores from 0.61 to 0.74 across the three classes. The model achieved an overall accuracy of 0.72 (72%), with macro average scores of 0.75 for precision, 0.72 for recall, and 0.72 for F1-score, while weighted averages are 0.78, 0.72, and 0.70, respectively, suggesting moderate performance with some variation across classes.

6.4. Gradient Boosting Classifier

Lastly, a confusion matrix analysis and a classification report of the Gradient Boosting Classifier model are also conducted. The implementation and respective outputs are as shown below.

```
pd.Series(GB_pred).value_counts()

CM(y_test, GB_pred, col_names=['Low', 'Medium', 'High'], title='- Gradient Boosting', cmap='Greens')

gb_accuracy = accuracy_score(y_test, GB_pred)
print(f'Overall Accuracy: {gb_accuracy:.3f}')
print(classification_report(y_test, GB_pred))
```

Figure 6.8: Implementation of Confusion Matrix and Classification Report of Gradient Boosting Classifier

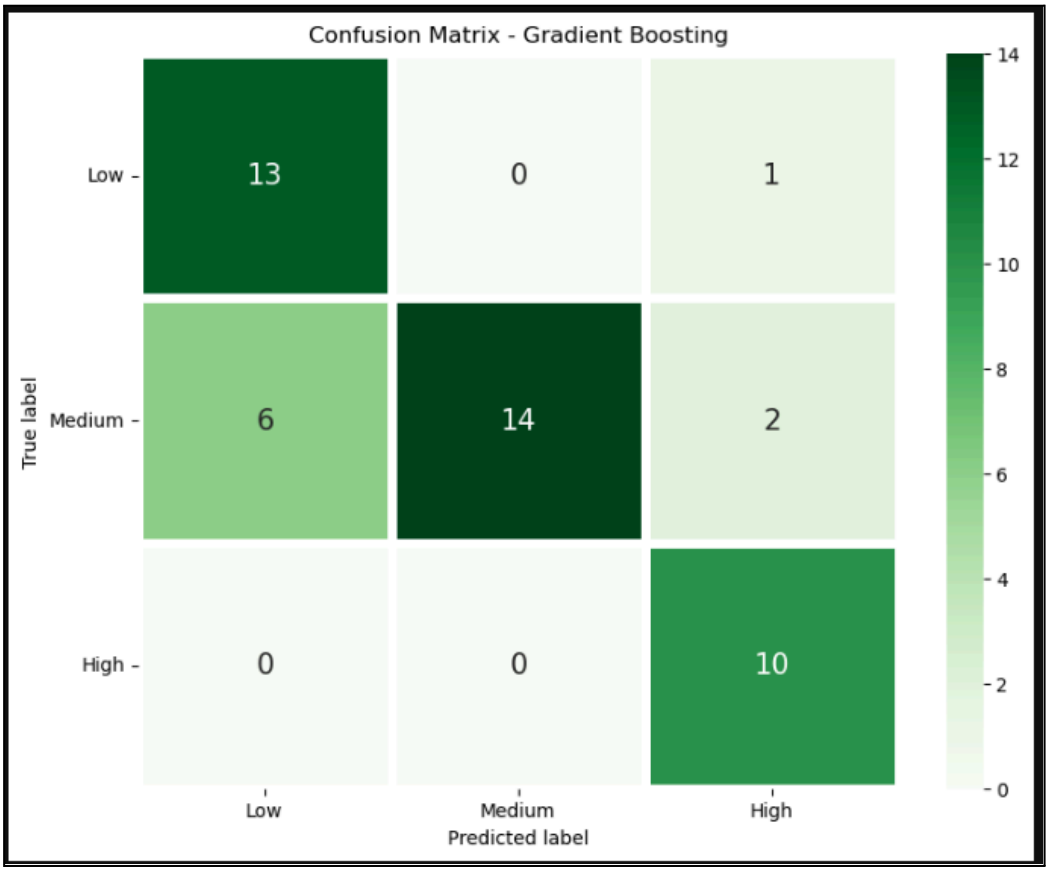

Figure 6.9: Confusion Matrix of Gradient Boosting Classifier

Overall Accuracy: 0.804

	precision	recall	f1-score	support
0	0.68	0.93	0.79	14
1	1.00	0.64	0.78	22
2	0.77	1.00	0.87	10
accuracy			0.80	46
macro avg	0.82	0.85	0.81	46
weighted avg	0.85	0.80	0.80	46

Figure 6.10: Classification Report of Gradient Boosting Classifier

According to the confusion matrix in Figure 6.9, the model exhibits strong classification performance. Out of 14 low cases, 13 were correctly classified while 1 was misclassified as high. For the 22 medium cases, 14 were accurately predicted, with 6 incorrectly classified as low and 2 as high. All 10 high cases were correctly identified with no misclassifications.

The classification report in Figure 6.10 reveals that the model achieved strong performance across all evaluation metrics. Precision values range from 0.68 to 1.00, recall from 0.64 to 1.00, and F1-scores from 0.78 to 0.87 across

the three classes. The model attained an overall accuracy of 0.80 (80%), with macro average scores of 0.82 for precision, 0.85 for recall, and 0.81 for F1-score, while weighted averages are 0.85, 0.80, and 0.80, respectively, demonstrating consistent and balanced performance across all classes.

6.5. Critical Analysis of the Results

Table 6.1: Comparison Table of Different Models (weighted average)

	Precision	**Recall**	**F1-Score**	**Accuracy**
Multinomial Logistic Regression	**0.92**	**0.91**	**0.91**	**0.91**
Random Forest Classifier	0.78	0.72	0.70	0.72
Gradient Boosting Classifier	0.85	0.80	0.80	0.80

As shown in Table 6.1 above, the Multinomial Logistic Regression model outperformed the other models from all evaluation metrics by achieving the highest precision of 0.92, recall 0.91, F1-score 0.91, and accuracy of 0.91. This explains that the model is very efficient in correctly classifying the 20 different variables in the dataset while maintaining a good stability between precision and recall.

In comparison, the Gradient Boosting Classifier achieved moderate and balanced results, with precision of 0.85, recall of 0.80, F1-score of 0.80, and accuracy of 0.80. While it is reasonably reliable but it is still considered less accurate compared to Multinomial Logistic Regression.

Next, Random Forest Classifier shows comparatively weaker performance, with precision of 0.78, recall of 0.72, F1-score of 0.70, and accuracy of 0.72. The lower recall and F1-score might indicate that the model struggles to correctly capture all the related variables from the dataset, leading to

more misclassifications. While Random Forest is generally robust for complex datasets, its lower performance in this case suggests that it may not be the most optimal model for the Lung Cancer dataset.

CHAPTER 7: CONCLUSION

This project successfully developed multiple models for lung cancer risk prediction using a dataset containing variance of variables. Through all the steps that we did such as data preprocessing, exploratory data analysis, and model development. We were able to identify relationships between risk factors and lung cancer levels. This analysis gave us the insights that variables such as chronic lung disease, genetic risk, alcohol use, air pollution, and coughing blood have strong relationships with higher cancer levels.

Three classification models were used during this project which are Multinomial Logistic Regression, Random Forest Classifier, and Gradient Boosting Classifier to train and test. The results showed us that Multinomial Logistic Regression achieved the best overall performance, with a precision of 0.92, recall of 0.91, F1-score of 0.91, and accuracy of 0.91. This strongly shows its strong and balanced capability to correctly identify lung cancer risk levels. Gradient Boosting Classifier also performed reasonably well, achieving an accuracy of 0.80, while Random Forest Classifier showed comparatively lower performance, with an accuracy of 0.72.

Based on these results, Multinomial Logistic Regression was selected as the most suitable model for this study. Besides its superior predictive performance, it also provides simplicity, faster training time, and higher interpretability through its coefficients. This interpretability is especially important in medical applications because it allows clinicians to better understand how individual risk factors contribute to different lung cancer risk levels.

Overall, this study showed us the potential of machine learning techniques in supporting lung cancer risk prediction and early detection. The findings are able to provide valuable information to assist future research in bioinformatics-driven healthcare.

REFERENCES

Alcohol and Cancer risk fact sheet. (2025, May 2). Cancer.gov. https://www.cancer.gov/about-cancer/causes-prevention/risk/alcohol/alcohol-fact-sheet

guslovesmath. (2024, March). *Lung Cancer ML Classification*. Kaggle.com; Kaggle. https://www.kaggle.com/code/guslovesmath/lung-cancer-ml-classification

Hammerschmidt, S., & Wirtz, H. (2009). Lung Cancer. *Deutsches Aerzteblatt Online*, *106*(49). https://doi.org/10.3238/arztebl.2009.0809

Langefeld, K. (2025, July 31). *World Lung Cancer Day - August 1, 2025 - Global Initiative for Chronic Obstructive Lung Disease - GOLD*. Global Initiative for Chronic Obstructive Lung Disease - GOLD. https://goldcopd.org/world-lung-cancer-day-august-1-2025/

Liao, W., Coupland, C., Burchardt, J., Baldwin, D. R., Gleeson, F., & Hippisley‐Cox, J. (2023). Predicting the future risk of lung cancer: development, and internal and external validation of the CanPredict (lung) model in 19·67 million people and evaluation of model performance against seven other risk prediction models. *The Lancet Respiratory Medicine*, *11*(8), 685–697. https://doi.org/10.1016/s2213-2600(23)00050-

Pacyna, J. M., & Pacyna, E. G. (2016). Environmental determinants of human health. In *Molecular and integrative toxicology*. https://doi.org/10.1007/978-3-319-43142-0

Performances analysis of heart disease dataset using different data mining classifications. (n.d.). https://scholar.google.com/citations?view_op=view_citation&hl=en&user=hueliFsAAAAJ&citation_for_view=hueliFsAAAAJ:kRWSkSYxWN8C

APPENDICES

Link GitHub Group 2: https://github.com/patyoonxin/SECB3203_25261.git