# Semester Project

# Probability & Statistics

## Lung Cancer Prediction

| Name | Roll No. | Section |
|---|---|---|
| Talha Rauf (GL) | 21F-9134 | BCS-5D |
| Danish Aziz | 21F-9229 | BCS-5D |
| Amash Rizwan | 21F-9116 | BCS-5D |
| Muhammad Faizan | 22F-3898 | BAI-3B |
| Muhammad Riyyan | 22F-3125 | BAI-3B |

# Problem Statement

We will analyze lung cancer, a prevalent cancer type, using this dataset to understand how various environmental and lifestyle factors affect individuals of different ages. With a focus on 21 factors, including **air pollution**, **alcohol use**, **dust allergy**, **smoking**, and **dry cough**, our goal is to identify the level of impact each factor has on lung cancer severity.

# Objective

Our main goal is to utilize statistical tools on this dataset to explore the relationships between various factors and their impact on lung cancer patients, specifically focusing on the severity of the disease. The following tools have been used to quantify specific relations:

- **Histograms**
- **Bar charts**
- **Pie charts**
- **Box plots**
- **Scatter plots**

And for analysis of this data set we have used:

- Basic descriptive statistics like (**mean**, **count** and **standard deviation**).

- Probability distribution (**Binomial distribution**) because for all the data, value of (**n**) was **less than 100**.
- **Logistic regression** because it is most helpful in determining disease like datasets.

## Data Description

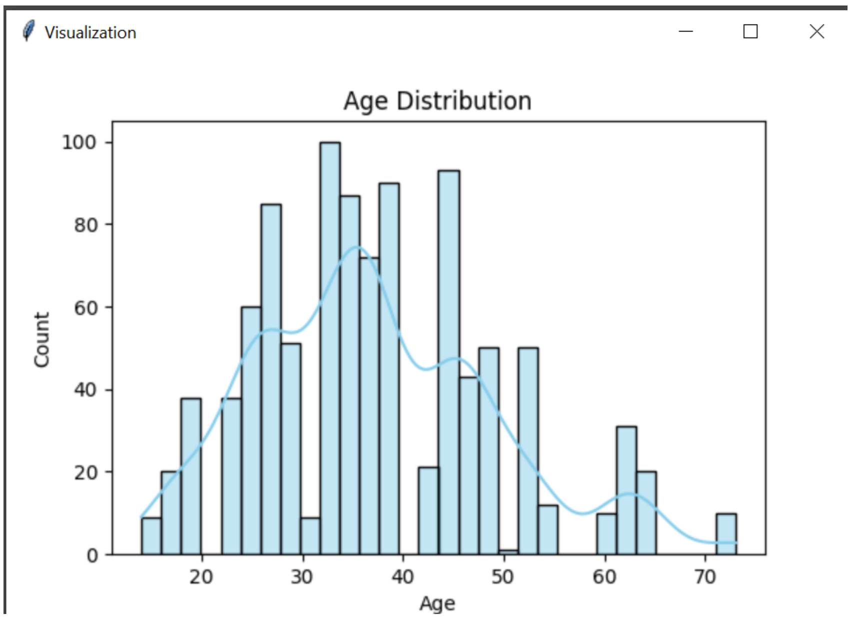
This dataset, titled **'Lung Cancer Prediction,'** is sourced from **Kaggle** and provides a detailed analysis of lung cancer patients. It covers a wide range of environmental and lifestyle factors that these patients are exposed to, and correlates them with the severity of their cancer, which is categorized as **'Level'** in the dataset. This comprehensive data aims to clarify the impact of various detrimental factors on lung cancer severity. The dataset can be accessed at the following link:

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

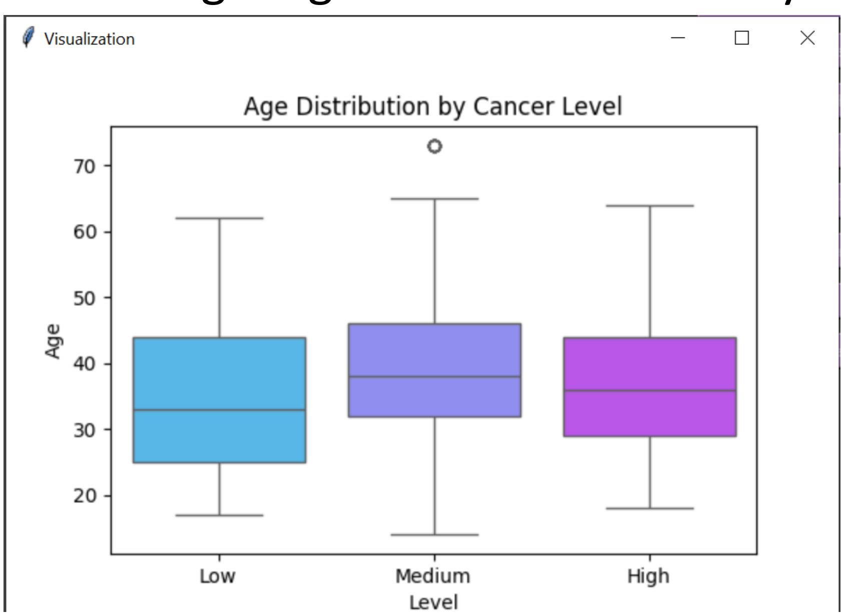
## Results

- The **User Interface:**
  (On next page)

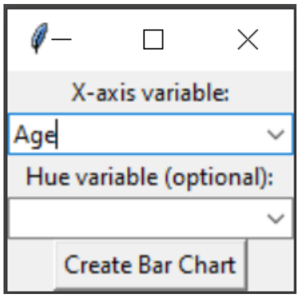

- For age distribution, we have a **Histogram:**

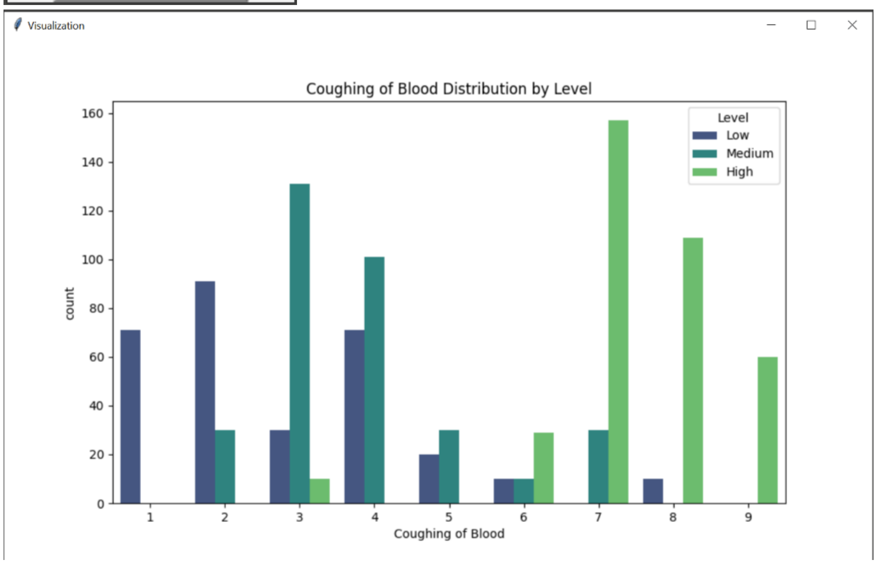


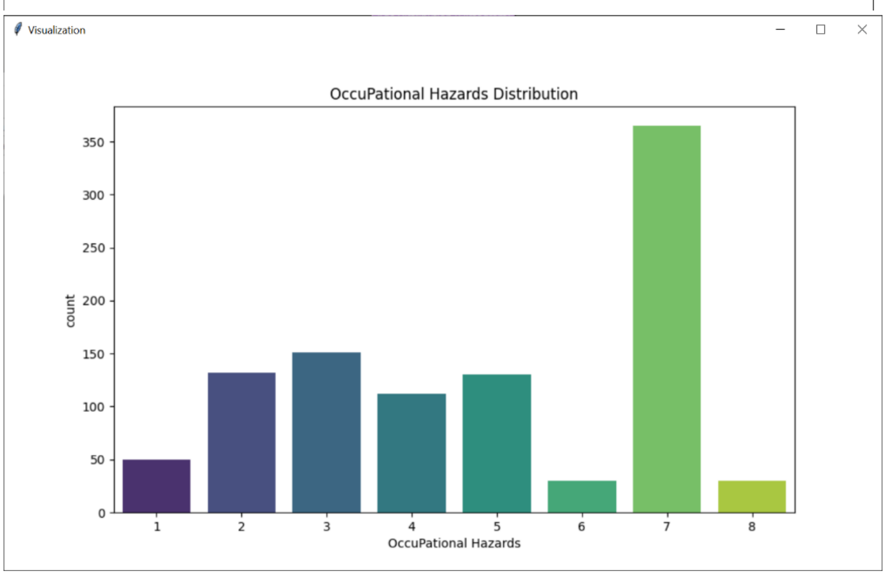

- Then we get age distribution level by **Box Plot:**



- Then we have **Bar Charts**, the prompt first asks for x axis parameter and then generates the bar chart**:**

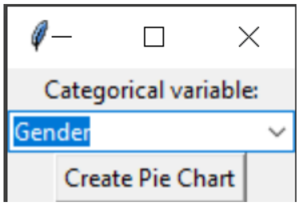


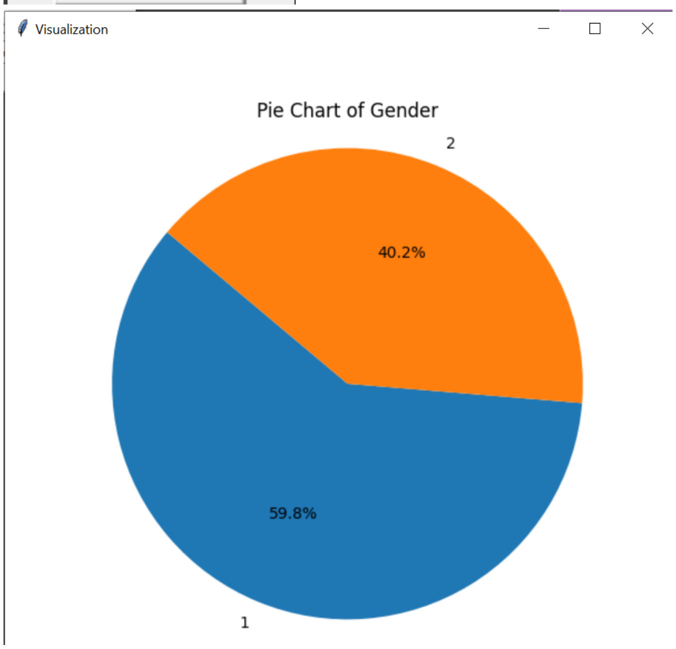



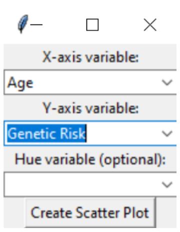

- Then we have **Pie Charts** in similar way**:**

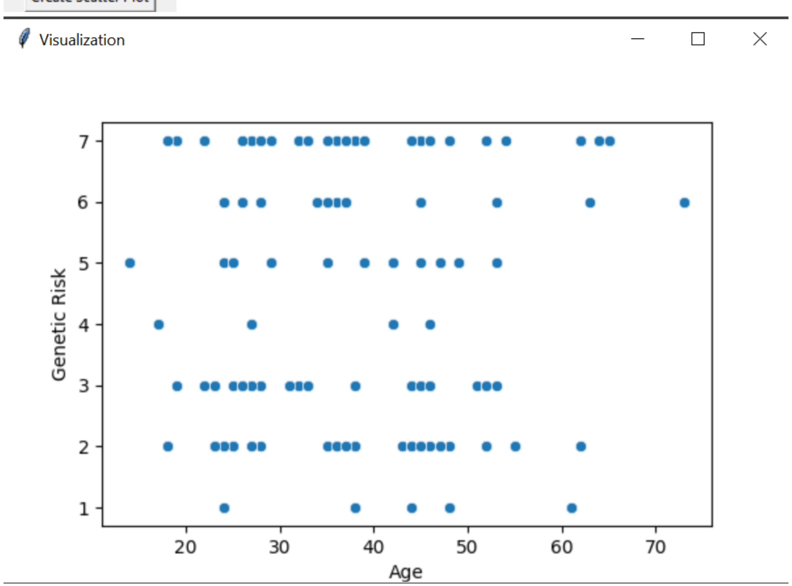
(On next page)



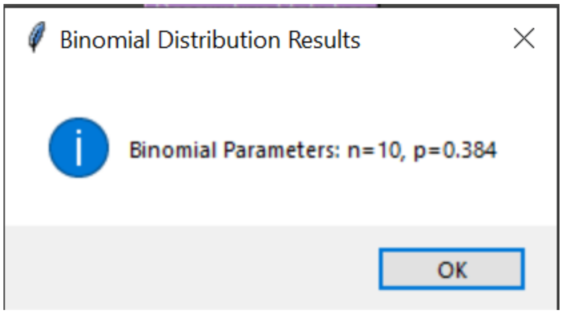




- Then we have **Scatter Plots** for comparison:

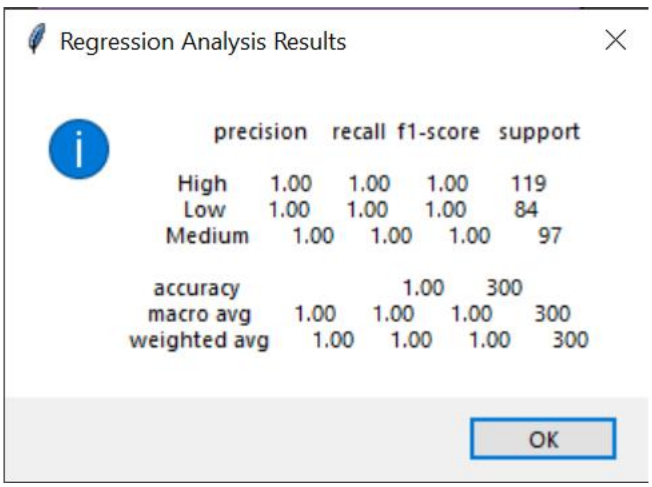


- Applying **Binomial distribution:**

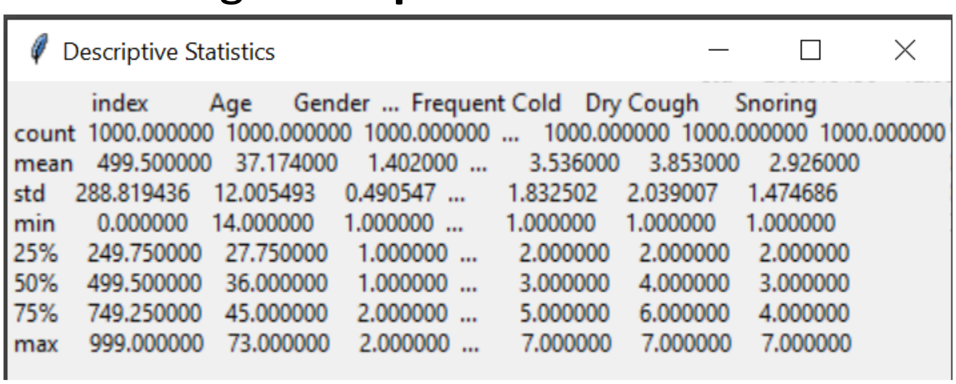



- Performing risk factor analysis using **Logistic Regression:**




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| High | 1.00 | 1.00 | 1.00 | 119 |
| Low | 1.00 | 1.00 | 1.00 | 84 |
| Medium | 1.00 | 1.00 | 1.00 | 97 |
| accuracy | | | 1.00 | 300 |
| macro avg | 1.00 | 1.00 | 1.00 | 300 |
| weighted avg | 1.00 | 1.00 | 1.00 | 300 |

- Concluding **Descriptive Statistics:**




| | index | Age | Gender | ... | Frequent Cold | Dry Cough | Snoring |
|---|---|---|---|---|---|---|---|
| count | 1000.000000 | 1000.000000 | 1000.000000 | ... | 1000.000000 | 1000.000000 | 1000.000000 |
| mean | 499.500000 | 37.174000 | 1.402000 | ... | 3.536000 | 3.853000 | 2.926000 |
| std | 288.819436 | 12.005493 | 0.490547 | ... | 1.832502 | 2.039007 | 1.474686 |
| min | 0.000000 | 14.000000 | 1.000000 | ... | 1.000000 | 1.000000 | 1.000000 |
| 25% | 249.750000 | 27.750000 | 1.000000 | ... | 2.000000 | 2.000000 | 2.000000 |
| 50% | 499.500000 | 36.000000 | 1.000000 | ... | 3.000000 | 4.000000 | 3.000000 |
| 75% | 749.250000 | 45.000000 | 2.000000 | ... | 5.000000 | 6.000000 | 4.000000 |
| max | 999.000000 | 73.000000 | 2.000000 | ... | 7.000000 | 7.000000 | 7.000000 |

# Codes

## UI.py

```
# This file sets up a GUI application for visualizing cancer patient data.

import tkinter as tk
from tkinter import ttk
import tkinter.messagebox
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from pandas import read_csv

# Importing functions from other modules for plotting and analysis
from visualization.histograms import create_age_histogram
from visualization.bar_charts import create_dynamic_bar_chart
from visualization.box_plots import create_age_distribution_by_level_chart
from visualization.scatter_plots import create_scatter_plot
from visualization.pie_charts import create_pie_chart
from analysis.probability_distributions import fit_binomial_distribution
from analysis.regression_models import perform_logistic_regression
from analysis.descriptive_stats import get_descriptive_stats

# Defining colors for the UI
bg_color = "#4f4f4e"
button_color = "#bd8dd6"
text_color = "#ebdcf2"

# Function to display plots in a new window
def display_plot(create_plot_func, data):
    plot_window = tk.Toplevel()
    plot_window.title("Visualization")
    fig = create_plot_func(data)
    canvas = FigureCanvasTkAgg(fig, master=plot_window)
    canvas.draw()
    canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

# Function to create input form for bar chart
def create_bar_chart_form(data):
    form_window = tk.Toplevel()
    form_window.title("Create Bar Chart")
    tk.Label(form_window, text="X-axis variable:").pack()
    x_var_entry = ttk.Combobox(form_window, values=list(data.columns))
    x_var_entry.pack()
    tk.Label(form_window, text="Hue variable (optional):").pack()
    hue_var_entry = ttk.Combobox(form_window, values=list(data.columns))
    hue_var_entry.pack()
    submit_button = tk.Button(form_window, text="Create Bar Chart",
                              command=lambda: display_plot(
                                  lambda data: create_dynamic_bar_chart(data,
x_var_entry.get(), hue_var_entry.get() if hue_var_entry.get() else None),
                                  data))
    submit_button.pack()

# Function to create input form for pie chart
def create_pie_chart_form(data):
    form_window = tk.Toplevel()
    form_window.title("Create Pie Chart")
    tk.Label(form_window, text="Categorical variable:").pack()
    cat_var_entry = ttk.Combobox(form_window, values=list(data.columns))
```

```
    cat_var_entry.pack()
    submit1_button = tk.Button(form_window, text="Create Pie Chart",
                               command=lambda: display_plot(
                                   lambda data: create_pie_chart(data, cat_var_entry.get()),
                                   data))
    submit1_button.pack()

# Function to create input form for scatter plot
def create_scatter_plot_form(data):
    form_window = tk.Toplevel()
    form_window.title("Create Scatter Plot")
    tk.Label(form_window, text="X-axis variable:").pack()
    x_var_entry = ttk.Combobox(form_window, values=list(data.columns))
    x_var_entry.pack()
    tk.Label(form_window, text="Y-axis variable:").pack()
    y_var_entry = ttk.Combobox(form_window, values=list(data.columns))
    y_var_entry.pack()
    tk.Label(form_window, text="Hue variable (optional):").pack()
    hue_var_entry = ttk.Combobox(form_window, values=list(data.columns))
    hue_var_entry.pack()
    submit_button = tk.Button(form_window, text="Create Scatter Plot",
                              command=lambda: display_plot(
                                  lambda data: create_scatter_plot(data, x_var_entry.get(),
y_var_entry.get(), hue_var_entry.get() if hue_var_entry.get() else None),
                                  data))
    submit_button.pack()

# Function to apply binomial distribution and display results
def on_apply_binomial():
    try:
        binomial_params = fit_binomial_distribution(data['Air Pollution'], trials=10,
success_prob=0.5)
        result_text = f"Binomial Parameters: n={binomial_params['n']},
p={binomial_params['p']:.3f}"
        tkinter.messagebox.showinfo("Binomial Distribution Results", result_text)
    except Exception as e:
        tkinter.messagebox.showerror("Error", str(e))

# Applying logistic regression
def on_perform_regression():
    try:
        # Applying predictors and target for logistic regression
        predictors = ['Age', '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']
        target = 'Level'

        # Perform logistic regression
        regression_result = perform_logistic_regression(data, predictors, target)

        # Displaying the regression result
        tkinter.messagebox.showinfo("Regression Analysis Results", regression_result)
    except Exception as e:
        tkinter.messagebox.showerror("Error", str(e))

# Applying descriptive statistics
def display_descriptive_stats(data):
```

```
    stats_window = tk.Toplevel()
    stats_window.title("Descriptive Statistics")
    stats_text = get_descriptive_stats(data)
    stats_label = tk.Label(stats_window, text=stats_text, justify="left")
    stats_label.pack()

# Loading the dataset
file_path = 'data/cancer patient data sets.csv'
data = read_csv(file_path)

# Setting up the main application window
root = tk.Tk()
root.title("Visualizations of Cancer Data")
root.configure(bg=bg_color) # for colors in UI

# Function to create styled buttons
def create_styled_button(parent, text, command):
    return tk.Button(parent, text=text, command=command, bg=button_color, fg=text_color)

# Creating and placing buttons for each visualization and analysis
button_age_distribution = create_styled_button(root, "Age Distribution (Histogram)", lambda:
display_plot(create_age_histogram, data))
button_age_distribution_by_level = create_styled_button(root, "Age Distribution by Level
(Box Plot)", lambda: display_plot(create_age_distribution_by_level_chart, data))
button_bar_chart = create_styled_button(root, "Create (Bar Charts)", lambda:
create_bar_chart_form(data))
button_gender_distribution = create_styled_button(root, "Create (Pie Charts)", lambda:
create_pie_chart_form(data))
button_scatter_plot = create_styled_button(root, "Create (Scatter Plots)", lambda:
create_scatter_plot_form(data))
binomial_button = create_styled_button(root, "Air Pollution (Binomial Distribution)",
on_apply_binomial)
regression_button = create_styled_button(root, "Perform Risk Factor Analysis (Logistic
Regression)", on_perform_regression)
button_descriptive_stats = create_styled_button(root, "Descriptive Statistics", lambda:
display_descriptive_stats(data))

# Calling button packs
button_age_distribution.pack(pady=5)
button_age_distribution_by_level.pack(pady=5)
button_bar_chart.pack(pady=5)
button_gender_distribution.pack(pady=5)
button_scatter_plot.pack(pady=5)
binomial_button.pack(pady=5)
regression_button.pack(pady=5)
button_descriptive_stats.pack(pady=5)

# Starting the main application loop
root.mainloop()
```

## Data-loader.py

```
import pandas as pd

def load_data(file_path):
    return pd.read_csv(file_path)
```

# Bar-chart.py

```
import seaborn as sns
from matplotlib.figure import Figure

def create_dynamic_bar_chart(data, x_var, hue_var=None):
    fig = Figure(figsize=(10, 6))
    ax = fig.subplots()
    sns.countplot(x=x_var, hue=hue_var, data=data, palette='viridis', ax=ax)
    ax.set_title(f'{x_var} Distribution' + (f' by {hue_var}' if hue_var else ''))
    return fig
```

# Box-plots.py

```
import seaborn as sns
from matplotlib.figure import Figure

def create_age_distribution_by_level_chart(data):
    fig = Figure(figsize=(6, 4))
    ax = fig.subplots()
    sns.boxplot(x='Level', y='Age', data=data, palette='cool', ax=ax)
    ax.set_title('Age Distribution by Cancer Level')
    return fig
```

# Histograms.py

```
import seaborn as sns
from matplotlib.figure import Figure

def create_age_histogram(data):
    fig = Figure(figsize=(6, 4))
    ax = fig.subplots()
    sns.histplot(data['Age'], bins=30, kde=True, color='skyblue', ax=ax)
    ax.set_title('Age Distribution')
    return fig
```

# Pie-charts.py

```
from matplotlib.figure import Figure

def create_pie_chart(data, column, title=None):
    fig = Figure(figsize=(6, 6))
    ax = fig.add_subplot(111)
    pie_data = data[column].value_counts()
    ax.pie(pie_data, labels=pie_data.index, autopct='%1.1f%%', startangle=140)
    ax.set_title(title if title else f'Pie Chart of {column}')
    ax.axis('equal')  # Equal aspect ratios
    return fig
```

# Scatter-plots.py

```
import seaborn as sns
from matplotlib.figure import Figure

def create_scatter_plot(data, x, y, hue=None, title=None):
    fig = Figure(figsize=(6, 4))
    ax = fig.subplots()
    sns.scatterplot(x=x, y=y, hue=hue, data=data, ax=ax)
    if title:
        ax.set_title(title)
    return fig
```

## Descriptive-stats.py

```
import pandas as pd

def get_descriptive_stats(data):
    numeric_data = data.select_dtypes(include=['number'])
    descriptive_stats = numeric_data.describe()
    return descriptive_stats

if __name__ == "__main__":
    # Loading the dataset
    file_path = 'data/cancer patient data sets.csv'
    data = pd.read_csv(file_path)

    # Printing descriptive statistics
    stats = get_descriptive_stats(data)
    print(stats)
```

## Probability-distributions.py

```
import scipy.stats as stats

def fit_binomial_distribution(data, trials, success_prob):
    successes = data.sum()
    estimated_p = successes / (trials * len(data))
    binom_dist = stats.binom(n=trials, p=estimated_p)
    return {"n": trials, "p": estimated_p, "distribution": binom_dist}
```

## Regression-models.py

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import pandas as pd

def perform_logistic_regression(data, predictors, target):
    # Preparing the data
    X = data[predictors]
    y = data[target]

    # Splitting the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    # Creating regression model
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    # Evaluating the model
    predictions = model.predict(X_test)
    report = classification_report(y_test, predictions)

    return report
```

# Conclusions

Based on the application of statistical analysis to our dataset, it has been observed that the majority of the subjects are identified as **Gender 1**, typically representing men. Additionally, factors such as **Air Pollution** and **Alcohol Use** demonstrate considerable variability among the patients. Similarly, variables like **Dust Allergy**, **Occupational Hazards**, and **Genetic Risk** also exhibit significant diversity across the dataset. A noteworthy correlation has been identified between **Air Pollution** and **Dust Allergy**, exhibiting a moderate positive relationship. This finding suggests a higher likelihood of **Dust Allergy** in patients exposed to elevated levels of **Air Pollution**. Furthermore, symptoms including **'Fatigue'**, **'Weight Loss'**, and **'Shortness of Breath'** display moderate correlation with each other, potentially indicating their prevalent co-occurrence in cancer patients. This analysis provides critical insights into the **environmental**, **genetic**, and **symptomatic** factors associated with **cancer**, underscoring the importance of these variables in understanding and managing the disease.