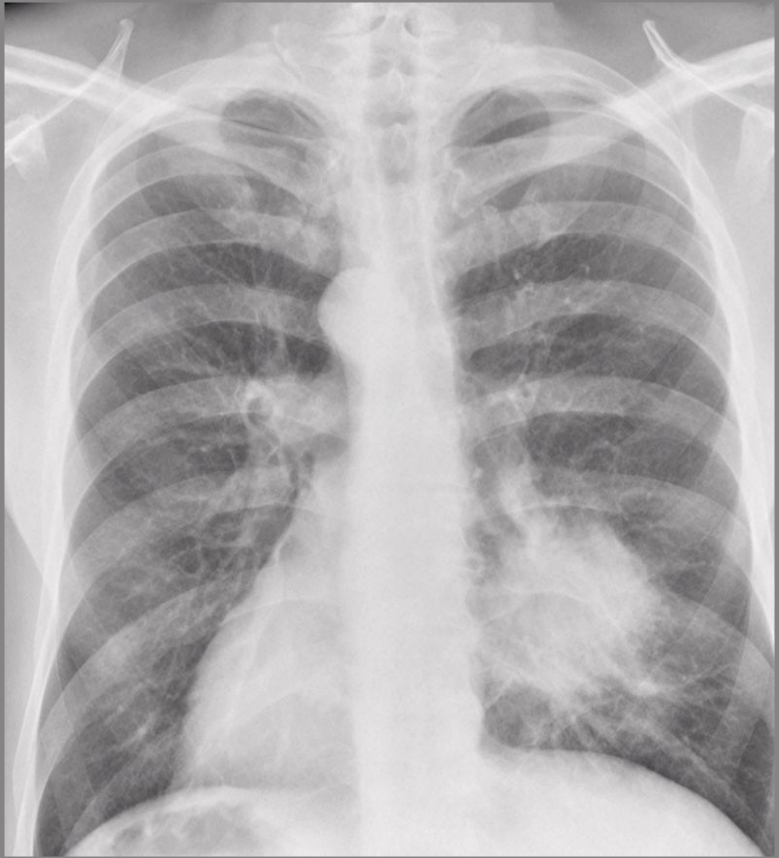
ST1(4483) Capstone Project

Lung Cancer

U3241507 Myeisha Jing Lin Foo



2023

Table of contents

Questions

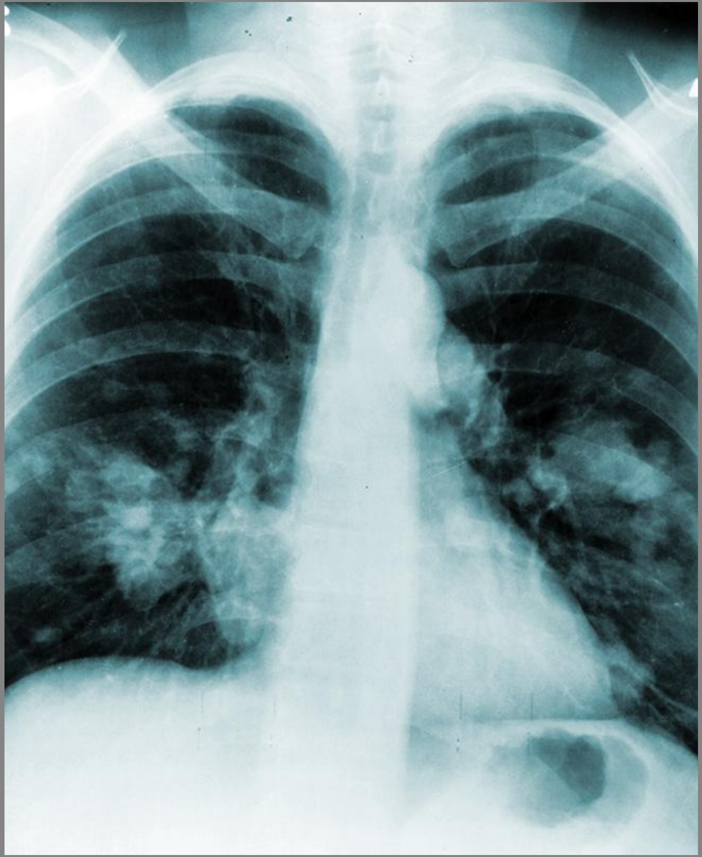
1. What is the distribution of lung cancer cases in the dataset?

2. How does age relate to the likelihood of lung cancer?

3. Are there any significant correlations between smoking and other attributes with the presence of lung cancer?

4. What are the most common symptoms and conditions associated with lung cancer?

5. Does Gender have correlation to lung cancer?



(01)

EDA

Exploratory Data Analysis

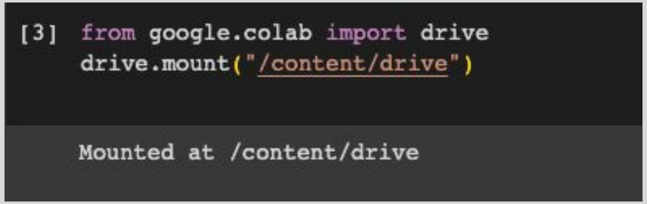
2022

Start

The following are the code snippets for dataset retrieval:

Import

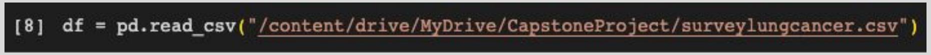
Preparing the environment to interact with Google Drive in Google Colab



```
[3] from google.colab import drive
    drive.mount("/content/drive")

    Mounted at /content/drive
```

2023

Dataset Retrieval

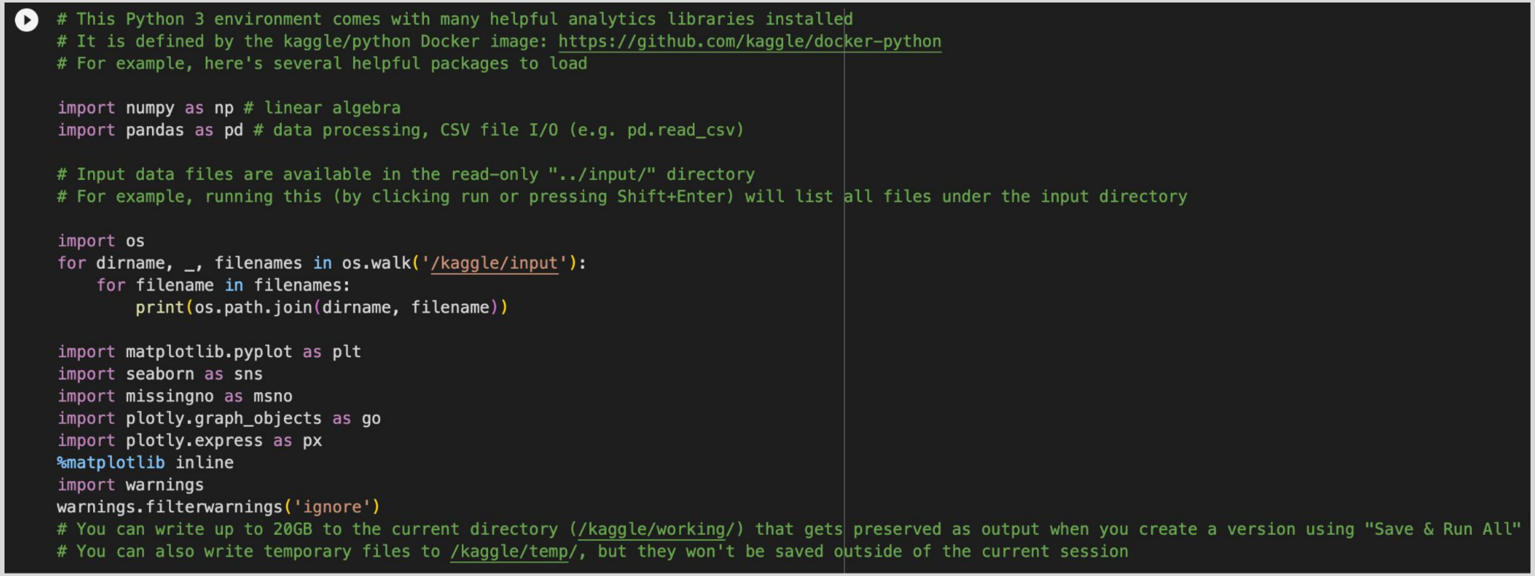


```
[8] df = pd.read_csv("/content/drive/MyDrive/CapstoneProject/surveylungcancer.csv")
```

To read the CSV (Comma-Separated Values) file named "surveylungcancer.csv" from a specific file path (My Drive)

Libraries

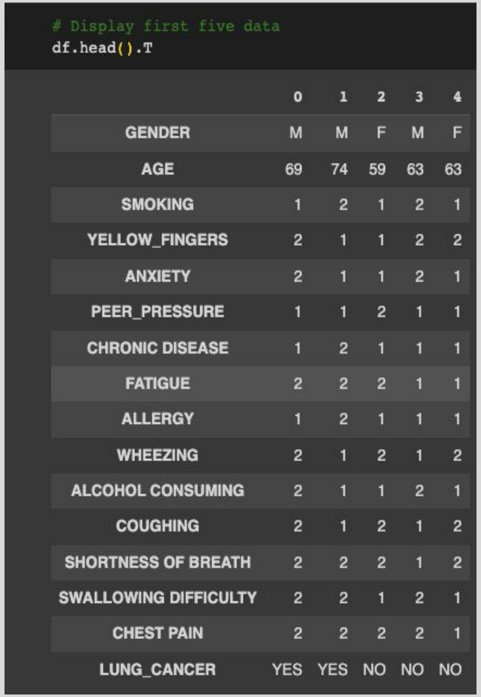
These are the libraries that are needed for my EDA.



```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import plotly.graph_objects as go
import plotly.express as px
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

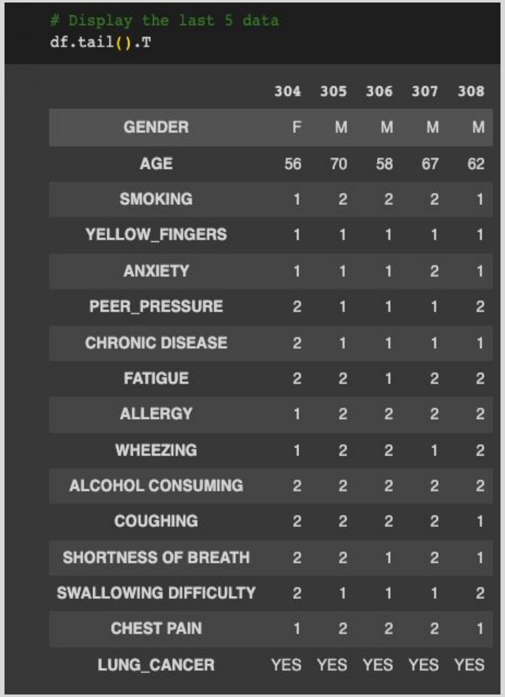
2023

2023



```
# Display first five data
df.head().T
```

	0	1	2	3	4
GENDER	M	M	F	M	F
AGE	69	74	59	63	63
SMOKING	1	2	1	2	1
YELLOW_FINGERS	2	1	1	2	2
ANXIETY	2	1	1	2	1
PEER_PRESSURE	1	1	2	1	1
CHRONIC DISEASE	1	2	1	1	1
FATIGUE	2	2	2	1	1
ALLERGY	1	2	1	1	1
WHEEZING	2	1	2	1	2
ALCOHOL CONSUMING	2	1	1	2	1
COUGHING	2	1	2	1	2
SHORTNESS OF BREATH	2	2	2	1	2
SWALLOWING DIFFICULTY	2	2	1	2	1
CHEST PAIN	2	2	2	2	1
LUNG_CANCER	YES	YES	NO	NO	NO



```
# Display the last 5 data
df.tail().T
```

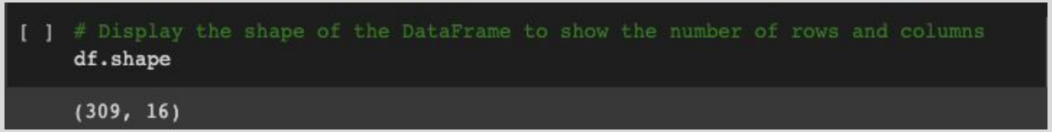
	304	305	306	307	308
GENDER	F	M	M	M	M
AGE	56	70	58	67	62
SMOKING	1	2	2	2	1
YELLOW_FINGERS	1	1	1	1	1
ANXIETY	1	1	1	2	1
PEER_PRESSURE	2	1	1	1	2
CHRONIC DISEASE	2	1	1	1	1
FATIGUE	2	2	1	2	2
ALLERGY	1	2	2	2	2
WHEEZING	1	2	2	1	2
ALCOHOL CONSUMING	2	2	2	2	2
COUGHING	2	2	2	2	1
SHORTNESS OF BREATH	2	2	1	2	1
SWALLOWING DIFFICULTY	2	1	1	1	2
CHEST PAIN	1	2	2	2	1
LUNG_CANCER	YES	YES	YES	YES	YES

Head & Tail

This displays the first 5 data description (Head) & the last 5 (Tails)

Shape of the DataFrame

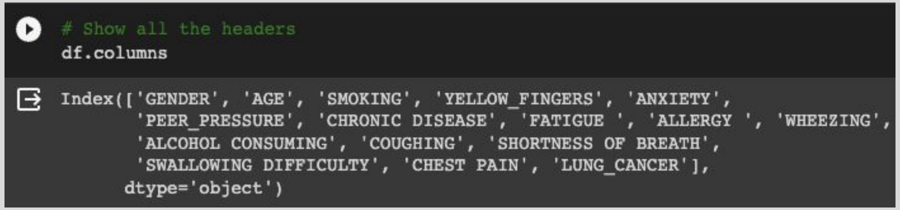
To determine and display the dimensions of the df, showing the number of rows and columns it contains. This provides an overview of the dataset's size.



```
[ ] # Display the shape of the DataFrame to show the number of rows and columns
    df.shape

    (309, 16)
```

Headers

To retrieve and display the column headers (or feature names) of the df. This step helps to identify the variables or

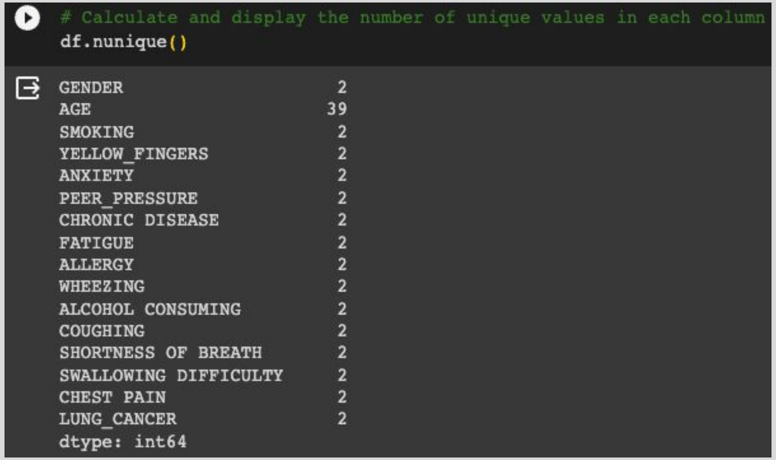


```
# Show all the headers
df.columns

Index(['GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY',
       'PEER_PRESSURE', 'CHRONIC DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING',
       'ALCOHOL CONSUMING', 'COUGHING', 'SHORTNESS OF BREATH',
       'SWALLOWING DIFFICULTY', 'CHEST PAIN', 'LUNG_CANCER'],
      dtype='object')
```

2023

Unique value & Information

2023

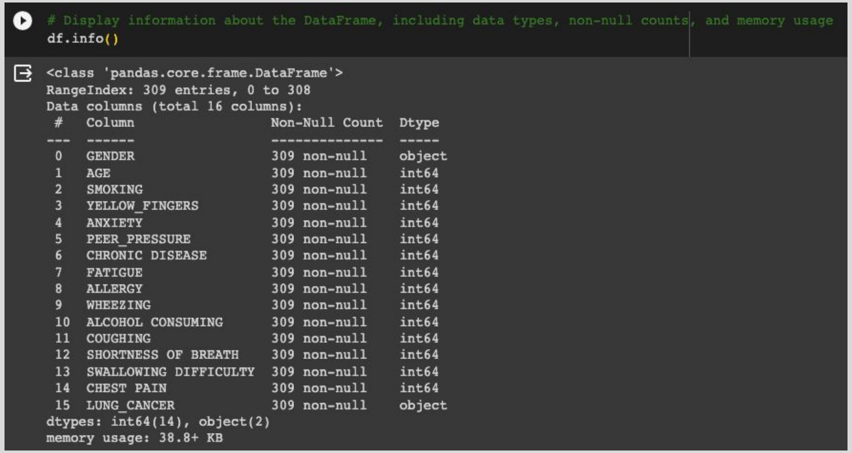


```
# Calculate and display the number of unique values in each column
df.nunique()
```

```
GENDER                    2
AGE                      39
SMOKING                   2
YELLOW_FINGERS            2
ANXIETY                   2
PEER_PRESSURE             2
CHRONIC DISEASE           2
FATIGUE                   2
ALLERGY                   2
WHEEZING                  2
ALCOHOL CONSUMING         2
COUGHING                  2
SHORTNESS OF BREATH       2
SWALLOWING DIFFICULTY     2
CHEST PAIN                2
LUNG_CANCER               2
dtype: int64
```

Unique values

To determine and display the count of unique values in each column of the df. This provides insights into the diversity of values in each feature or attribute.



```
# Display information about the DataFrame, including data types, non-null counts, and memory usage
df.info()
```

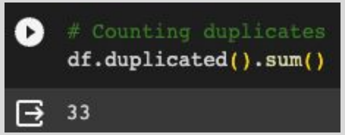
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   GENDER                 309 non-null    object
 1   AGE                    309 non-null    int64
 2   SMOKING                309 non-null    int64
 3   YELLOW_FINGERS         309 non-null    int64
 4   ANXIETY                309 non-null    int64
 5   PEER_PRESSURE          309 non-null    int64
 6   CHRONIC DISEASE        309 non-null    int64
 7   FATIGUE                309 non-null    int64
 8   ALLERGY                309 non-null    int64
 9   WHEEZING               309 non-null    int64
 10  ALCOHOL CONSUMING      309 non-null    int64
 11  COUGHING               309 non-null    int64
 12  SHORTNESS OF BREATH    309 non-null    int64
 13  SWALLOWING DIFFICULTY  309 non-null    int64
 14  CHEST PAIN             309 non-null    int64
 15  LUNG_CANCER            309 non-null    object
dtypes: int64(14), object(2)
memory usage: 38.8+ KB
```

Information

To provide comprehensive information about the df, including the data types of each column, the number of non-null (non-missing) values, and the memory usage. This summary is helpful for understanding the structure and content of the dataset

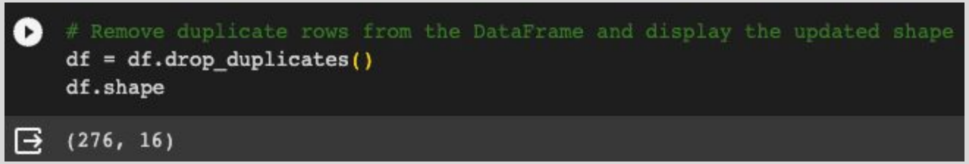
Duplicate & Checking

2023



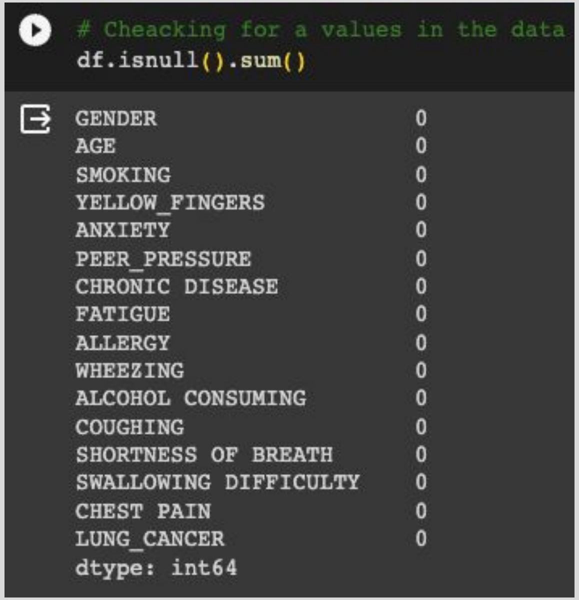
```
# Counting duplicates
df.duplicated().sum()
```

```
33
```



```
# Remove duplicate rows from the DataFrame and display the updated shape
df = df.drop_duplicates()
df.shape
```

```
(276, 16)
```



```
# Cheacking for a values in the data
df.isnull().sum()
```

```
GENDER                   0
AGE                      0
SMOKING                  0
YELLOW_FINGERS           0
ANXIETY                  0
PEER_PRESSURE            0
CHRONIC DISEASE          0
FATIGUE                  0
ALLERGY                  0
WHEEZING                 0
ALCOHOL CONSUMING        0
COUGHING                 0
SHORTNESS OF BREATH      0
SWALLOWING DIFFICULTY    0
CHEST PAIN               0
LUNG_CANCER              0
dtype: int64
```

Duplicate

To count and display the number of duplicate rows in the df & to remove duplicate rows from the df and then display the updated shape of the DataFrame, showing the number of rows and columns after removing duplicates.

Checking

To examine the df for missing (null) values in each column and calculate the sum of missing values in each column.

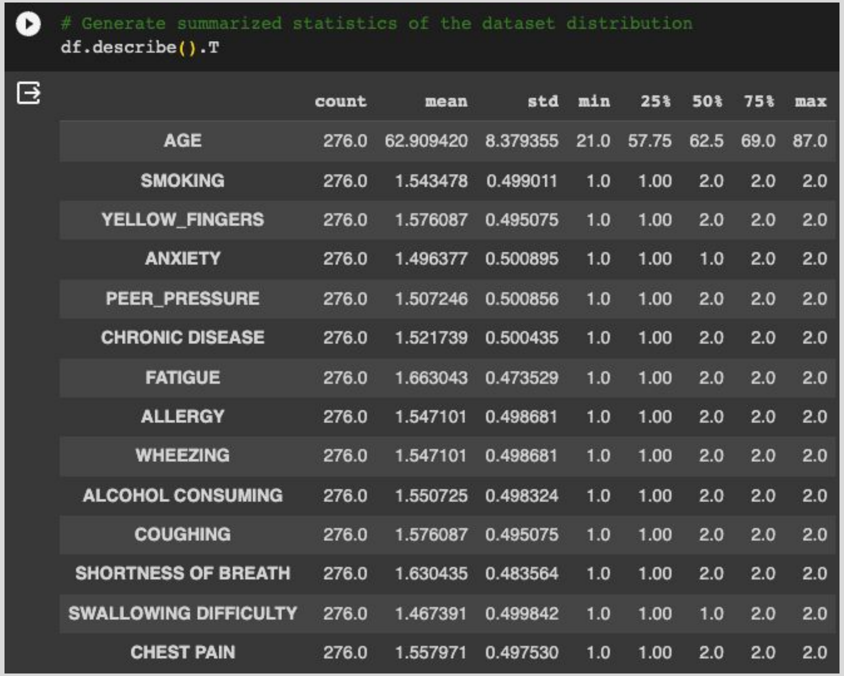
Summarized statistics

2023

To generate and display summary statistics of the dataset's distribution.

The describe() method provides statistics such as count, mean, standard deviation, minimum, and maximum for each numerical column in the df.

Transposing the result with .T makes it easier to read when there are many columns.

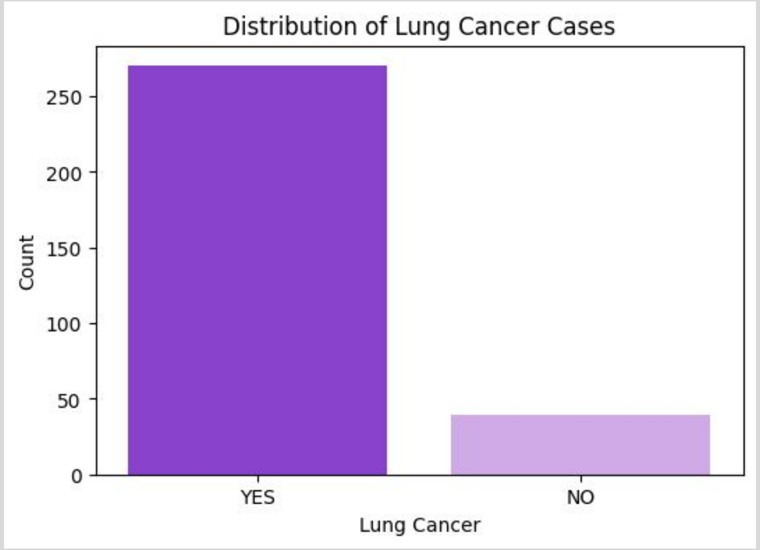


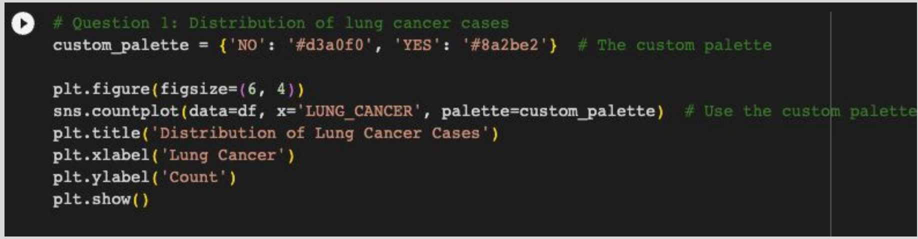
```
# Generate summarized statistics of the dataset distribution
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
AGE	276.0	62.909420	8.379355	21.0	57.75	62.5	69.0	87.0
SMOKING	276.0	1.543478	0.499011	1.0	1.00	2.0	2.0	2.0
YELLOW_FINGERS	276.0	1.576087	0.495075	1.0	1.00	2.0	2.0	2.0
ANXIETY	276.0	1.496377	0.500895	1.0	1.00	1.0	2.0	2.0
PEER_PRESSURE	276.0	1.507246	0.500856	1.0	1.00	2.0	2.0	2.0
CHRONIC DISEASE	276.0	1.521739	0.500435	1.0	1.00	2.0	2.0	2.0
FATIGUE	276.0	1.663043	0.473529	1.0	1.00	2.0	2.0	2.0
ALLERGY	276.0	1.547101	0.498681	1.0	1.00	2.0	2.0	2.0
WHEEZING	276.0	1.547101	0.498681	1.0	1.00	2.0	2.0	2.0
ALCOHOL CONSUMING	276.0	1.550725	0.498324	1.0	1.00	2.0	2.0	2.0
COUGHING	276.0	1.576087	0.495075	1.0	1.00	2.0	2.0	2.0
SHORTNESS OF BREATH	276.0	1.630435	0.483564	1.0	1.00	2.0	2.0	2.0
SWALLOWING DIFFICULTY	276.0	1.467391	0.499842	1.0	1.00	1.0	2.0	2.0
CHEST PAIN	276.0	1.557971	0.497530	1.0	1.00	2.0	2.0	2.0

1. What is the distribution of lung cancer cases in the dataset?

2023



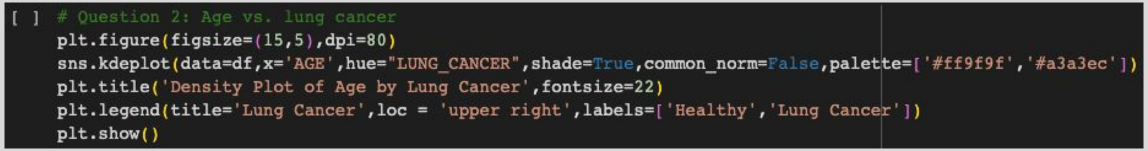


```
# Question 1: Distribution of lung cancer cases
custom_palette = {'NO': '#d3a0f0', 'YES': '#8a2be2'}  # The custom palette

plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='LUNG_CANCER', palette=custom_palette)  # Use the custom palette
plt.title('Distribution of Lung Cancer Cases')
plt.xlabel('Lung Cancer')
plt.ylabel('Count')
plt.show()
```

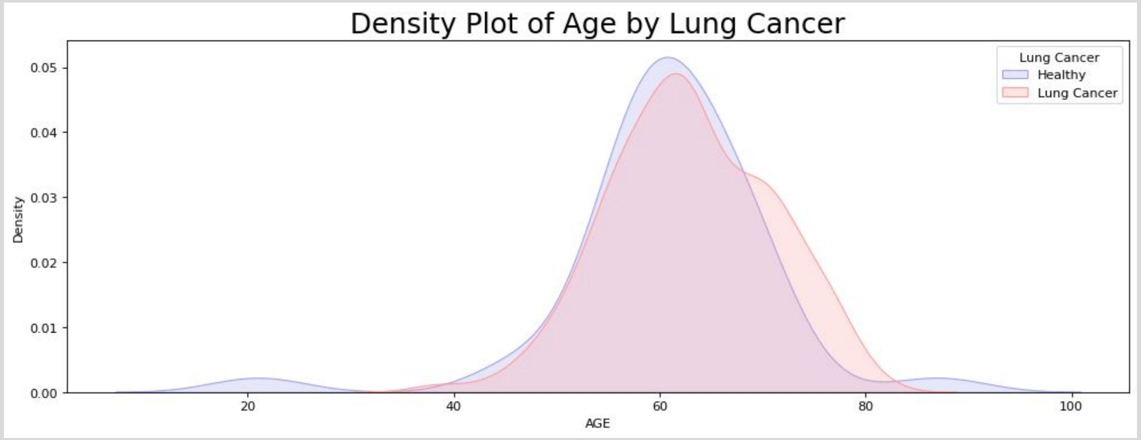
Finding

In the dataset there are more people with lung cancer than without

2. How does age relate to the likelihood of lung cancer?



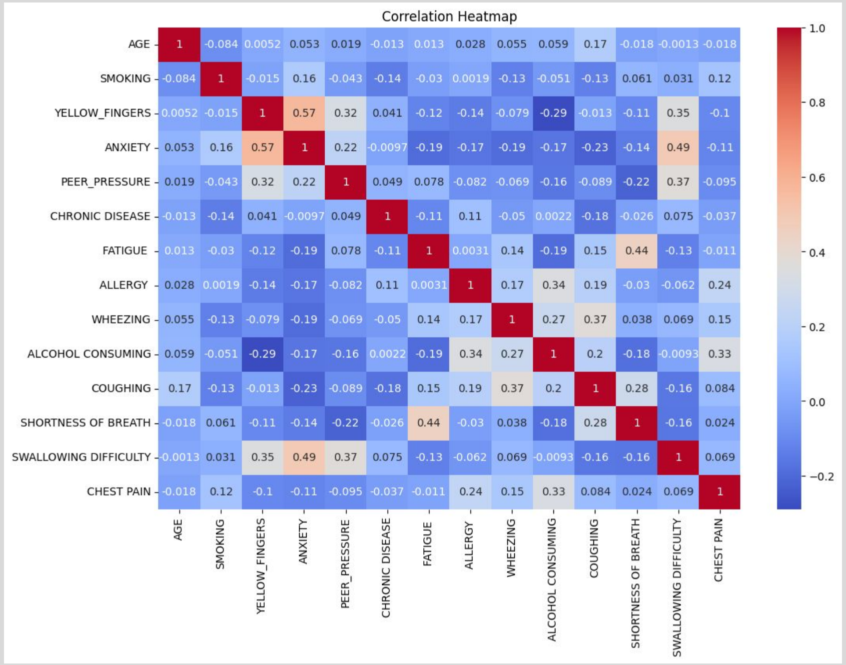
```
[ ]  # Question 2: Age vs. lung cancer
     plt.figure(figsize=(15,5),dpi=80)
     sns.kdeplot(data=df,x='AGE',hue="LUNG_CANCER",shade=True,common_norm=False,palette=['#ff9f9f','#a3a3ec'])
     plt.title('Density Plot of Age by Lung Cancer',fontsize=22)
     plt.legend(title='Lung Cancer',loc = 'upper right',labels=['Healthy','Lung Cancer'])
     plt.show()
```



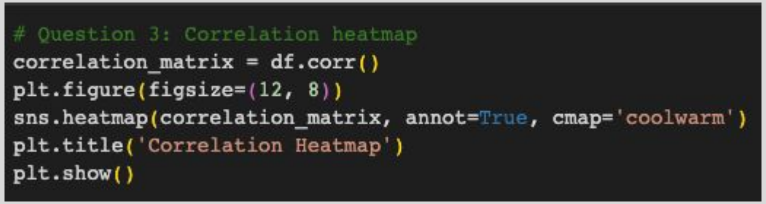
Finding

- People who are typically in the age 20 range tend to have healthy lungs.
- The age ranges between 30 to 90 it can vary whether you have a healthy or unhealthy lungs.

3. Are there any significant correlations between smoking and other attributes with the presence of lung cancer?



	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN
AGE	1	-0.084	0.0052	0.053	0.019	-0.013	0.013	0.028	0.055	0.059	0.17	-0.018	-0.0013	-0.018
SMOKING	-0.084	1	-0.015	0.16	-0.043	-0.14	-0.03	0.0019	-0.13	-0.051	-0.13	0.061	0.031	0.12
YELLOW_FINGERS	0.0052	-0.015	1	0.57	0.32	0.041	-0.12	-0.14	-0.079	-0.29	-0.013	-0.11	0.35	-0.1
ANXIETY	0.053	0.16	0.57	1	0.22	-0.0097	-0.19	-0.17	-0.19	-0.17	-0.23	-0.14	0.49	-0.11
PEER_PRESSURE	0.019	-0.043	0.32	0.22	1	0.049	0.078	-0.082	-0.069	-0.16	-0.089	-0.22	0.37	-0.095
CHRONIC DISEASE	-0.013	-0.14	0.041	-0.0097	0.049	1	-0.11	0.11	-0.05	0.0022	-0.18	-0.026	0.075	-0.037
FATIGUE	0.013	-0.03	-0.12	-0.19	0.078	-0.11	1	0.0031	0.14	-0.19	0.15	0.44	-0.13	-0.011
ALLERGY	0.028	0.0019	-0.14	-0.17	-0.082	0.11	0.0031	1	0.17	0.34	0.19	-0.03	-0.062	0.24
WHEEZING	0.055	-0.13	-0.079	-0.19	-0.069	-0.05	0.14	0.17	1	0.27	0.37	0.038	0.069	0.15
ALCOHOL CONSUMING	0.059	-0.051	-0.29	-0.17	-0.16	0.0022	-0.19	0.34	0.27	1	0.2	-0.18	-0.0093	0.33
COUGHING	0.17	-0.13	-0.013	-0.23	-0.089	-0.18	0.15	0.19	0.37	0.2	1	0.28	-0.16	0.084
SHORTNESS OF BREATH	-0.018	0.061	-0.11	-0.14	-0.22	-0.026	0.44	-0.03	0.038	-0.18	0.28	1	-0.16	0.024
SWALLOWING DIFFICULTY	-0.0013	0.031	0.35	0.49	0.37	0.075	-0.13	-0.062	0.069	-0.0093	-0.16	-0.16	1	0.069
CHEST PAIN	-0.018	0.12	-0.1	-0.11	-0.095	-0.037	-0.011	0.24	0.15	0.33	0.084	0.024	0.069	1



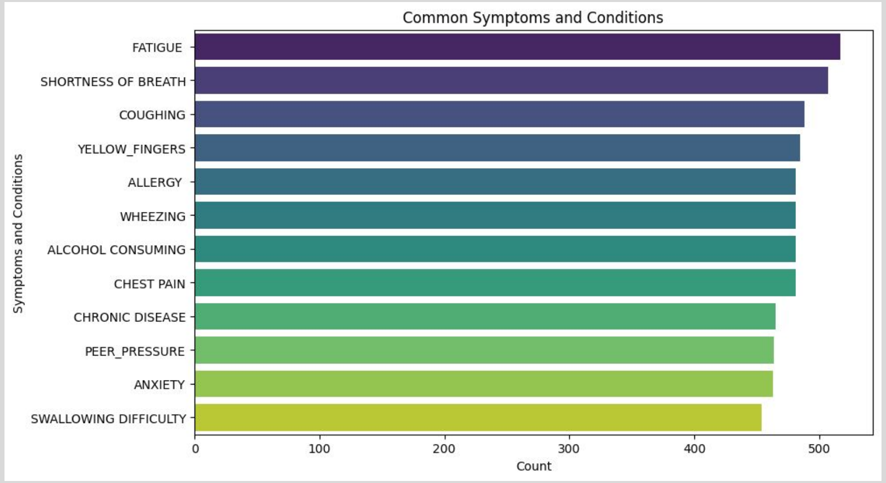
```python
# Question 3: Correlation heatmap
correlation_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

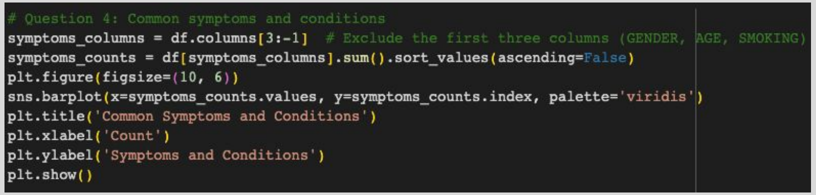
2023

Finding

- According to the heat correlation map majourity of the symptons and conditions have no correlation to one another.
- With the exception of Anxiety and Yellow fingers.

4. What are the most common symptoms and conditions associated with lung cancer?





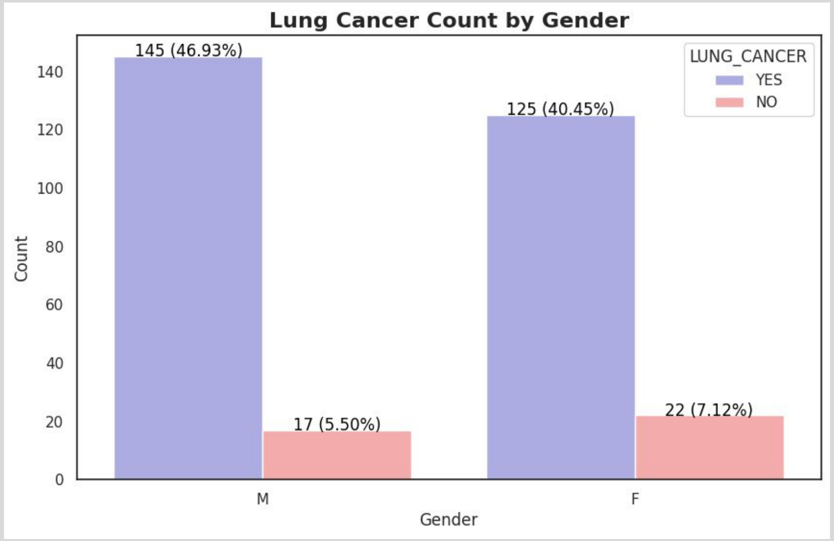
```
# Question 4: Common symptoms and conditions
symptoms_columns = df.columns[3:-1]  # Exclude the first three columns (GENDER, AGE, SMOKING)
symptoms_counts = df[symptoms_columns].sum().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=symptoms_counts.values, y=symptoms_counts.index, palette='viridis')
plt.title('Common Symptoms and Conditions')
plt.xlabel('Count')
plt.ylabel('Symptoms and Conditions')
plt.show()
```

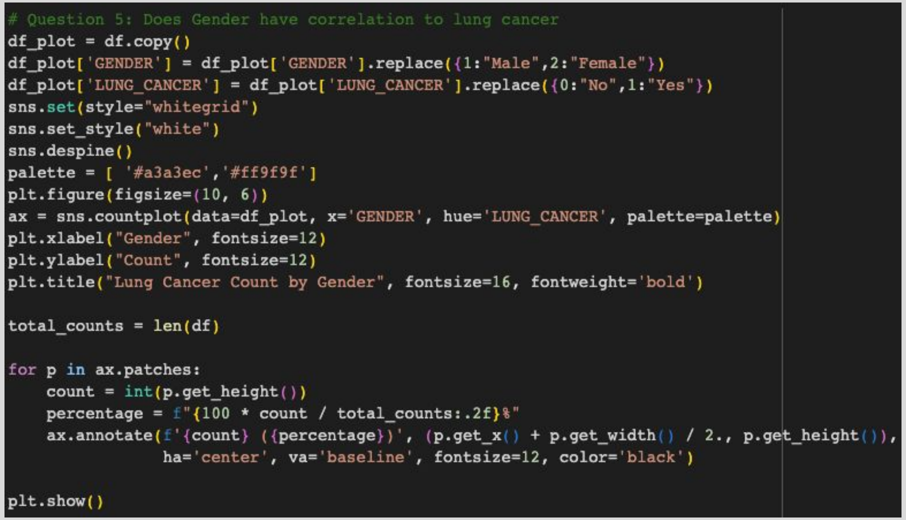
2023

Finding

- The most common symptom and condition is Fatigue , Shortness Of Breath and Coughing.
- The least common symptom and condition is Peer Pressure , Anxiety and Swallowing Difficulty.

5. Does Gender have correlation to lung cancer





```python
# Question 5: Does Gender have correlation to lung cancer
df_plot = df.copy()
df_plot['GENDER'] = df_plot['GENDER'].replace({1:"Male",2:"Female"})
df_plot['LUNG_CANCER'] = df_plot['LUNG_CANCER'].replace({0:"No",1:"Yes"})
sns.set(style="whitegrid")
sns.set_style("white")
sns.despine()
palette = [ '#a3a3ec','#ff9f9f']
plt.figure(figsize=(10, 6))
ax = sns.countplot(data=df_plot, x='GENDER', hue='LUNG_CANCER', palette=palette)
plt.xlabel("Gender", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.title("Lung Cancer Count by Gender", fontsize=16, fontweight='bold')

total_counts = len(df)

for p in ax.patches:
    count = int(p.get_height())
    percentage = f"{100 * count / total_counts:.2f}%"
    ax.annotate(f'{count} ({percentage})', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='baseline', fontsize=12, color='black')

plt.show()
```

2023

Finding

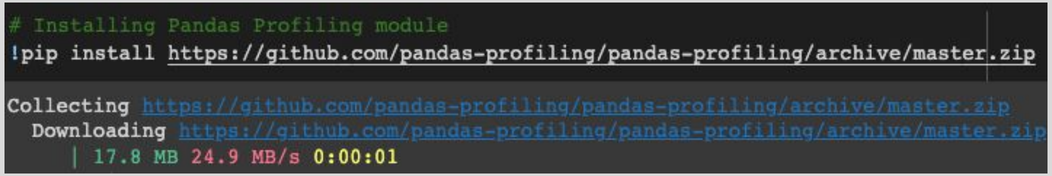
- Gender does not have correlation to having or not having lung cancer
- It shows 46.93% of men have lung cancer and 40.45% of women have lung cancer, having a 6.48% difference.
- It shows 5.50% of men do not have lung cancer and 7.12% of women do not have lung cancer, having a 2.48% difference.

2023

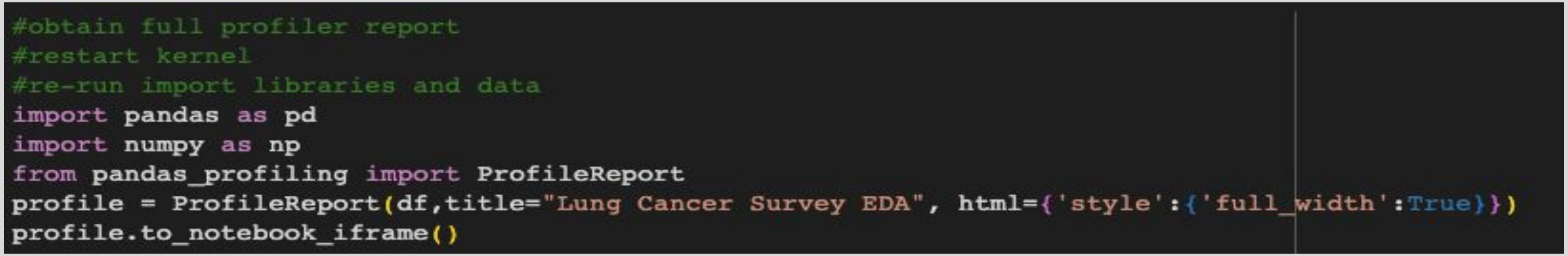
Pandas

Install the Pandas Profiling module

Generates a full profiler report using the Pandas Profiling library



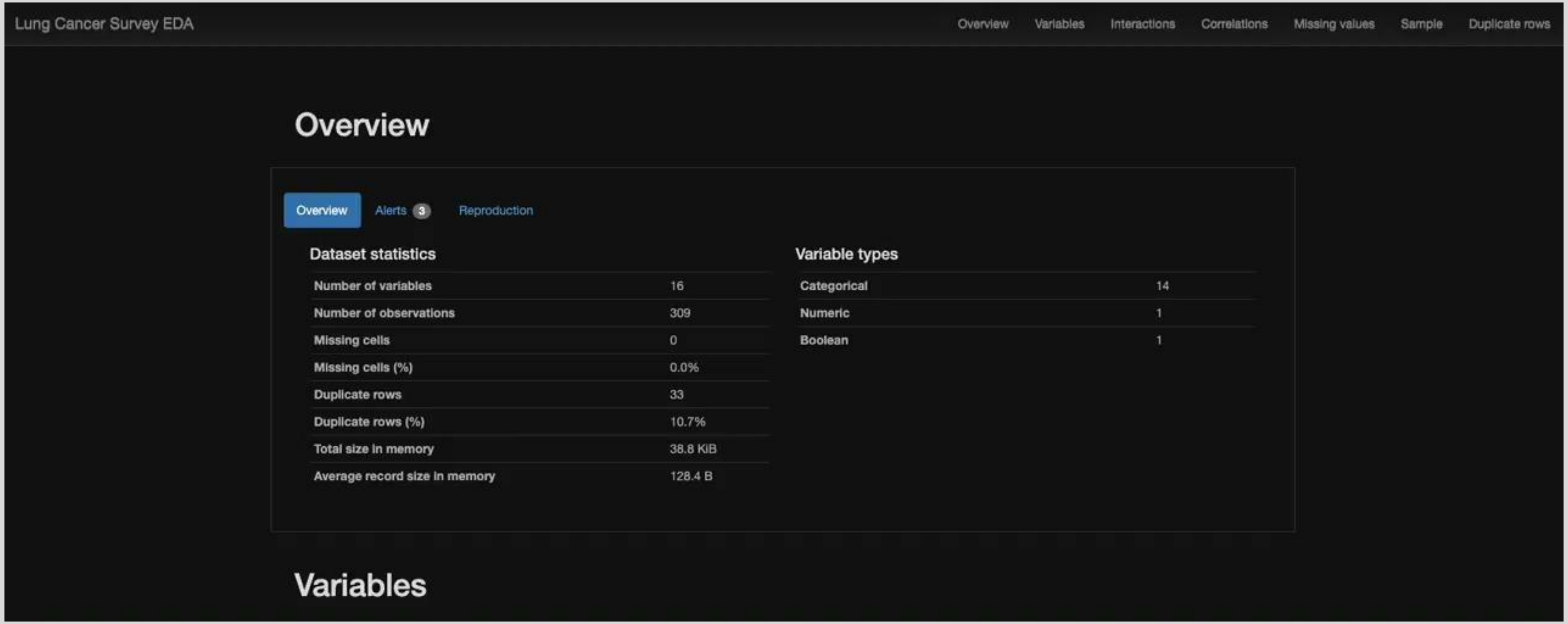
```
# Installing Pandas Profiling module
!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip

Collecting https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
  Downloading https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
     | 17.8 MB 24.9 MB/s 0:00:01
```



```
#obtain full profiler report
#restart kernel
#re-run import libraries and data
import pandas as pd
import numpy as np
from pandas_profiling import ProfileReport
profile = ProfileReport(df,title="Lung Cancer Survey EDA", html={'style':{'full_width':True}})
profile.to_notebook_iframe()
```

Profiler Report

Here is the generated profile report



Overview

Dataset statistics

Number of variables	16
Number of observations	309
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	33
Duplicate rows (%)	10.7%
Total size in memory	38.8 KiB
Average record size in memory	128.4 B

Variable types

Categorical	14
Numeric	1
Boolean	1

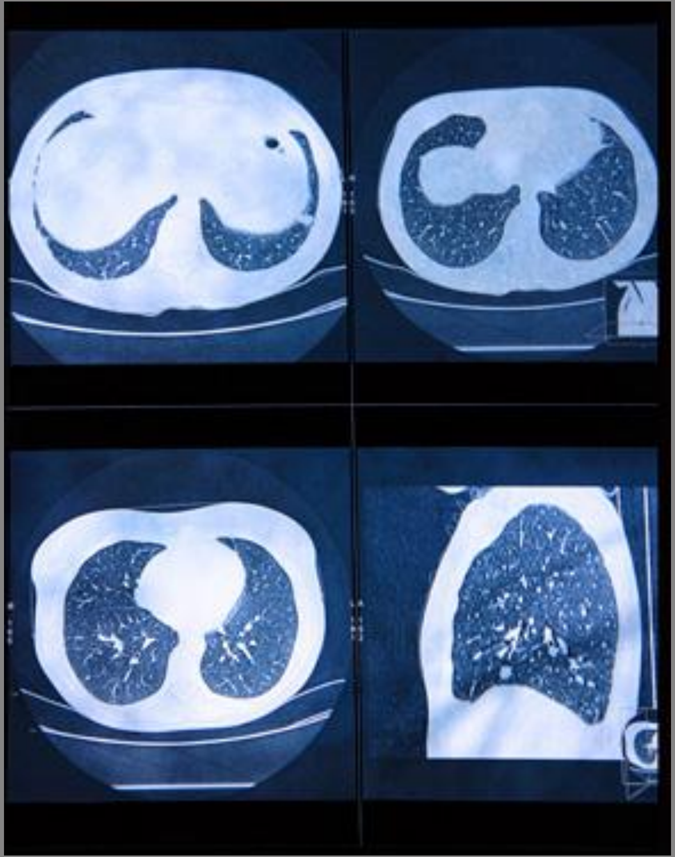
Variables

2023

(02)

PDA

Predictive Analysis Task

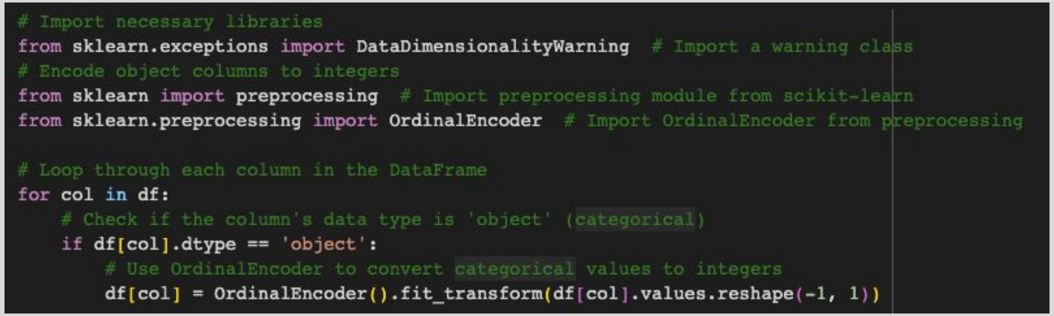


2022

Import

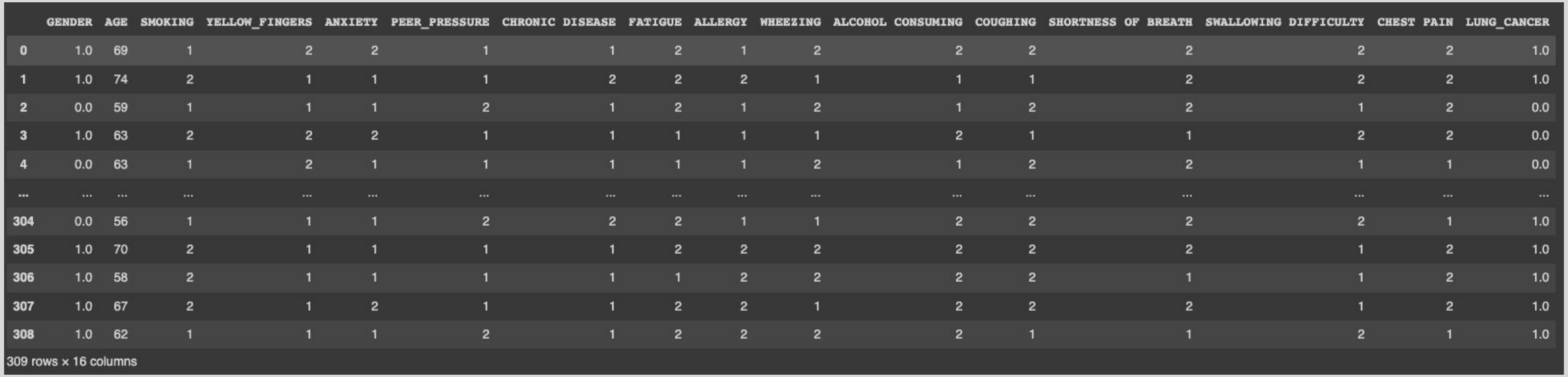
Preprocess a DataFrame by converting categorical (object) values into numerical representations

Required for machine learning algorithms to work with categorical data, as many algorithms expect numerical input



```python
# Import necessary libraries
from sklearn.exceptions import DataDimensionalityWarning  # Import a warning class
# Encode object columns to integers
from sklearn import preprocessing  # Import preprocessing module from scikit-learn
from sklearn.preprocessing import OrdinalEncoder  # Import OrdinalEncoder from preprocessing

# Loop through each column in the DataFrame
for col in df:
    # Check if the column's data type is 'object' (categorical)
    if df[col].dtype == 'object':
        # Use OrdinalEncoder to convert categorical values to integers
        df[col] = OrdinalEncoder().fit_transform(df[col].values.reshape(-1, 1))
```

2023

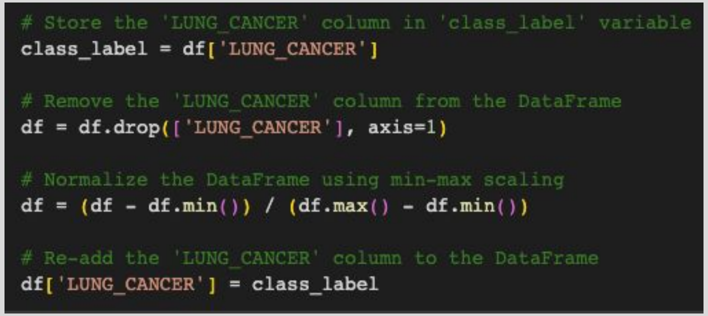


	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	1.0	69	1	2	2	1	1	2	1	2	2	2	2	2	2	1.0
1	1.0	74	2	1	1	1	2	2	2	1	1	1	2	2	2	1.0
2	0.0	59	1	1	1	2	1	2	1	2	1	2	2	1	2	0.0
3	1.0	63	2	2	2	1	1	1	1	1	2	1	1	2	2	0.0
4	0.0	63	1	2	1	1	1	1	1	2	1	2	2	1	1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
304	0.0	56	1	1	1	2	2	2	1	1	2	2	2	2	1	1.0
305	1.0	70	2	1	1	1	1	2	2	2	2	2	2	1	2	1.0
306	1.0	58	2	1	1	1	1	1	2	2	2	2	1	1	2	1.0
307	1.0	67	2	1	2	1	1	2	2	1	2	2	2	1	2	1.0
308	1.0	62	1	1	1	2	1	2	2	2	2	1	1	2	1	1.0

309 rows × 16 columns

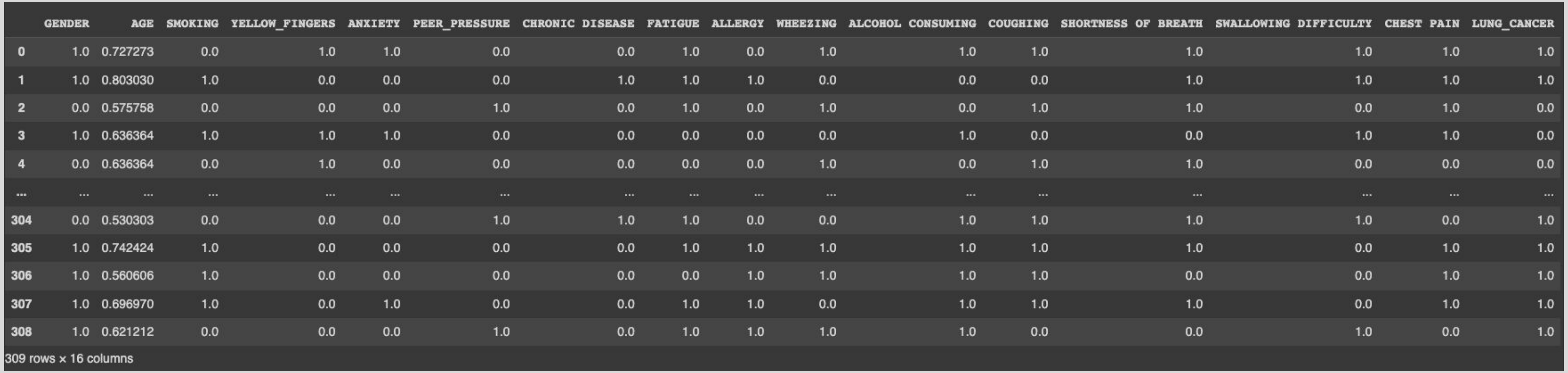
Min-Max

It separates the target variable, scales the feature variables using min-max scaling, and then combines the target variable back with the scaled features to prepare the data for modeling



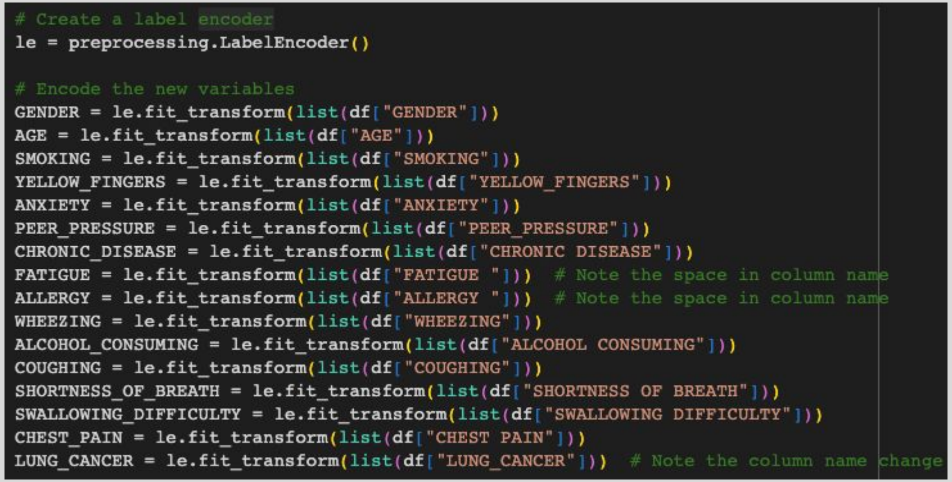
```python
# Store the 'LUNG_CANCER' column in 'class_label' variable
class_label = df['LUNG_CANCER']

# Remove the 'LUNG_CANCER' column from the DataFrame
df = df.drop(['LUNG_CANCER'], axis=1)

# Normalize the DataFrame using min-max scaling
df = (df - df.min()) / (df.max() - df.min())

# Re-add the 'LUNG_CANCER' column to the DataFrame
df['LUNG_CANCER'] = class_label
```

2023



	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	1.0	0.727273	0.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	0.803030	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0
2	0.0	0.575758	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0
3	1.0	0.636364	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0
4	0.0	0.636364	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
304	0.0	0.530303	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0
305	1.0	0.742424	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0
306	1.0	0.560606	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0
307	1.0	0.696970	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0
308	1.0	0.621212	0.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0

309 rows × 16 columns



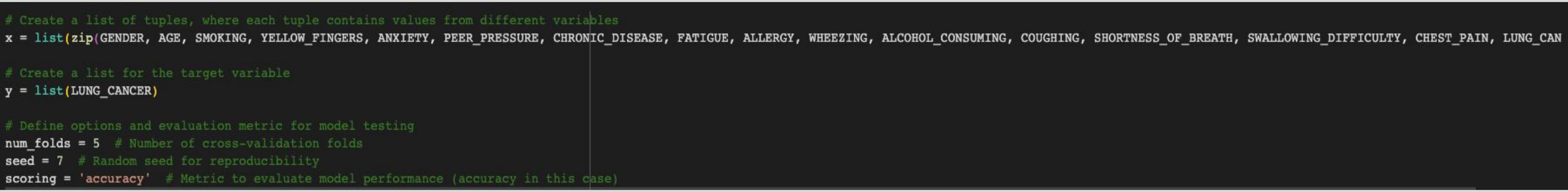
```python
# Create a label encoder
le = preprocessing.LabelEncoder()

# Encode the new variables
GENDER = le.fit_transform(list(df["GENDER"]))
AGE = le.fit_transform(list(df["AGE"]))
SMOKING = le.fit_transform(list(df["SMOKING"]))
YELLOW_FINGERS = le.fit_transform(list(df["YELLOW_FINGERS"]))
ANXIETY = le.fit_transform(list(df["ANXIETY"]))
PEER_PRESSURE = le.fit_transform(list(df["PEER_PRESSURE"]))
CHRONIC_DISEASE = le.fit_transform(list(df["CHRONIC DISEASE"]))
FATIGUE = le.fit_transform(list(df["FATIGUE "]))  # Note the space in column name
ALLERGY = le.fit_transform(list(df["ALLERGY "]))  # Note the space in column name
WHEEZING = le.fit_transform(list(df["WHEEZING"]))
ALCOHOL_CONSUMING = le.fit_transform(list(df["ALCOHOL CONSUMING"]))
COUGHING = le.fit_transform(list(df["COUGHING"]))
SHORTNESS_OF_BREATH = le.fit_transform(list(df["SHORTNESS OF BREATH"]))
SWALLOWING_DIFFICULTY = le.fit_transform(list(df["SWALLOWING DIFFICULTY"]))
CHEST_PAIN = le.fit_transform(list(df["CHEST PAIN"]))
LUNG_CANCER = le.fit_transform(list(df["LUNG_CANCER"]))  # Note the column name change
```

2023

Label

The purpose of this label encoding is to prepare the categorical variables for use in machine learning algorithms

Many machine learning models require numerical inputs, and label encoding is one way to achieve this



```
# Create a list of tuples, where each tuple contains values from different variables
x = list(zip(GENDER, AGE, SMOKING, YELLOW_FINGERS, ANXIETY, PEER_PRESSURE, CHRONIC_DISEASE, FATIGUE, ALLERGY, WHEEZING, ALCOHOL_CONSUMING, COUGHING, SHORTNESS_OF_BREATH, SWALLOWING_DIFFICULTY, CHEST_PAIN, LUNG_CAN

# Create a list for the target variable
y = list(LUNG_CANCER)

# Define options and evaluation metric for model testing
num_folds = 5  # Number of cross-validation folds
seed = 7  # Random seed for reproducibility
scoring = 'accuracy'  # Metric to evaluate model performance (accuracy in this case)
```

2023

List

Creating a List for the Target Variable y = list

- The target variable represents what you want to predict or classify using the machine learning model

Creating a List of Tuples x = list(zip

These tuples are used as the feature variables for a machine learning model.

The combination of these variables as features allows the model to learn patterns and relationships in the data to make predictions

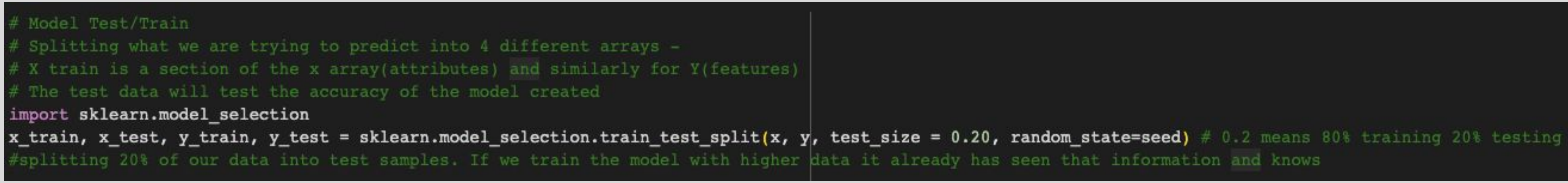
Defining Options and Evaluation Metric

num_folds = 5: This sets the number of cross-validation folds Cross-validation is a technique used to assess the model's performance

seed = 7: This defines a random seed, which is used to control the randomization process during cross-validation and other random operations

scoring = 'accuracy': This specifies the evaluation metric to measure the model's performance

assess the model's accuracy in making predictions



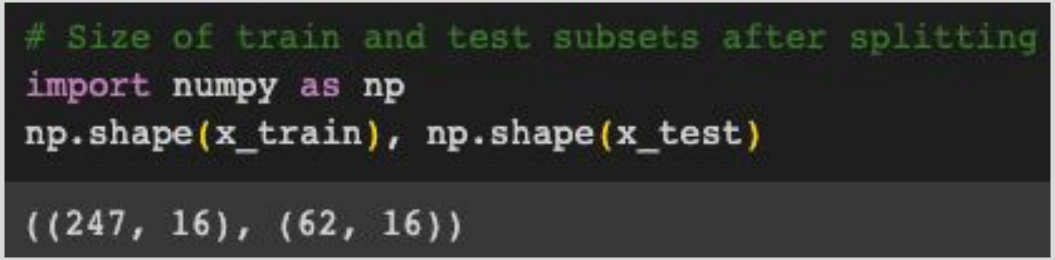
```
# Model Test/Train
# Splitting what we are trying to predict into 4 different arrays -
# X train is a section of the x array(attributes) and similarly for Y(features)
# The test data will test the accuracy of the model created
import sklearn.model_selection
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y, test_size = 0.20, random_state=seed) # 0.2 means 80% training 20% testing
#splitting 20% of our data into test samples. If we train the model with higher data it already has seen that information and knows
```

2023

Splitting

The purpose of splitting the data into training and testing sets is to evaluate how well the machine learning model generalizes to unseen data.

This helps avoid issues like overfitting, where a model memorizes the training data but performs poorly on new data



```
# Size of train and test subsets after splitting
import numpy as np
np.shape(x_train), np.shape(x_test)
```

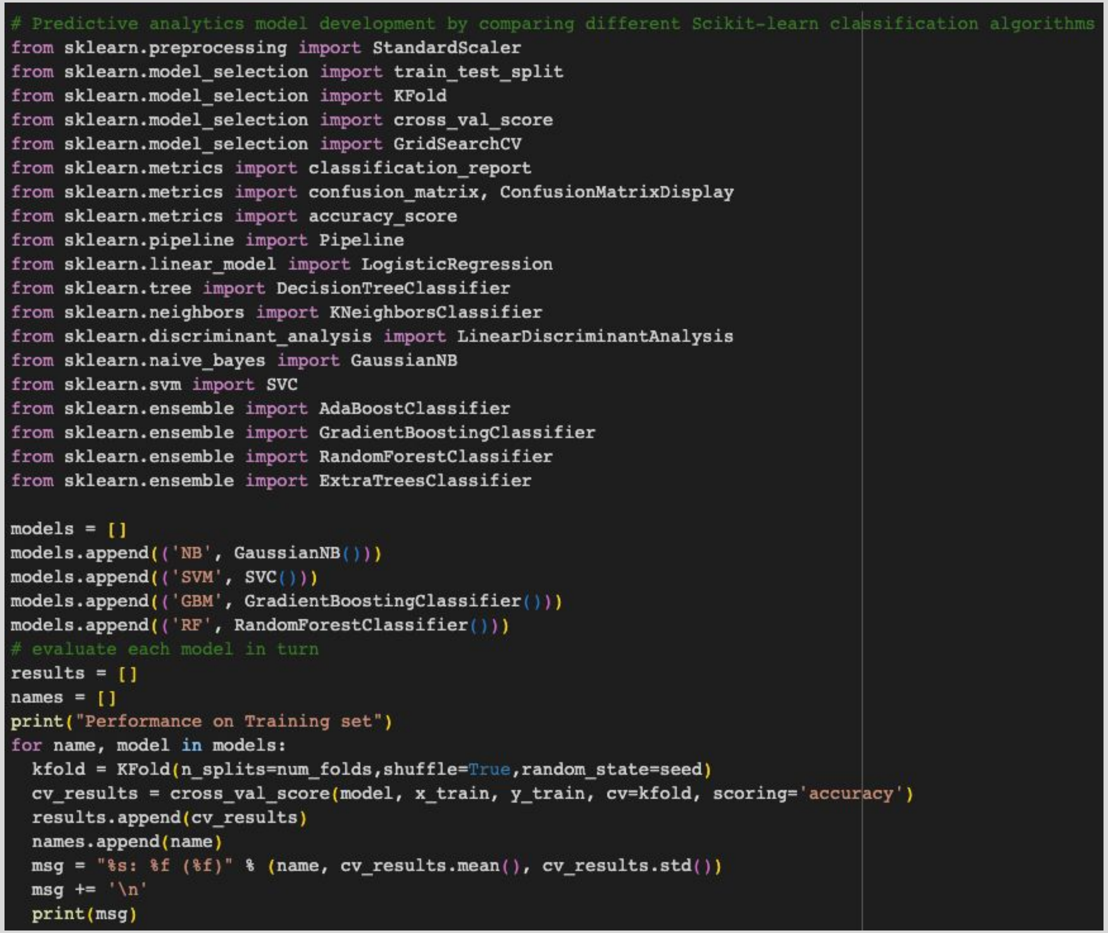
```
((247, 16), (62, 16))
```

2023

Size

The purpose of calculating and printing the shapes of these subsets is to help you verify that the data splitting process has occurred as intended

2023

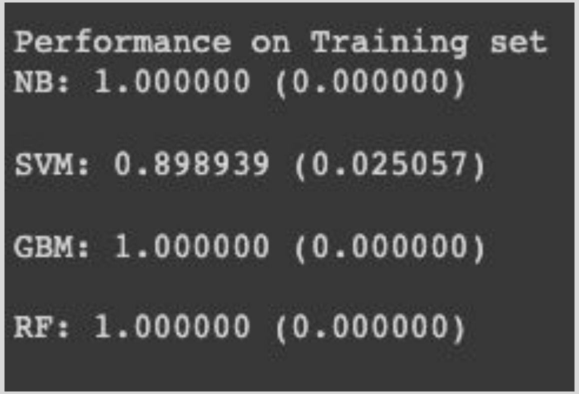


```
# Predictive analytics model development by comparing different Scikit-learn classification algorithms
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

models = []
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('GBM', GradientBoostingClassifier()))
models.append(('RF', RandomForestClassifier()))
# evaluate each model in turn
results = []
names = []
print("Performance on Training set")
for name, model in models:
  kfold = KFold(n_splits=num_folds,shuffle=True,random_state=seed)
  cv_results = cross_val_score(model, x_train, y_train, cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  msg += '\n'
  print(msg)
```

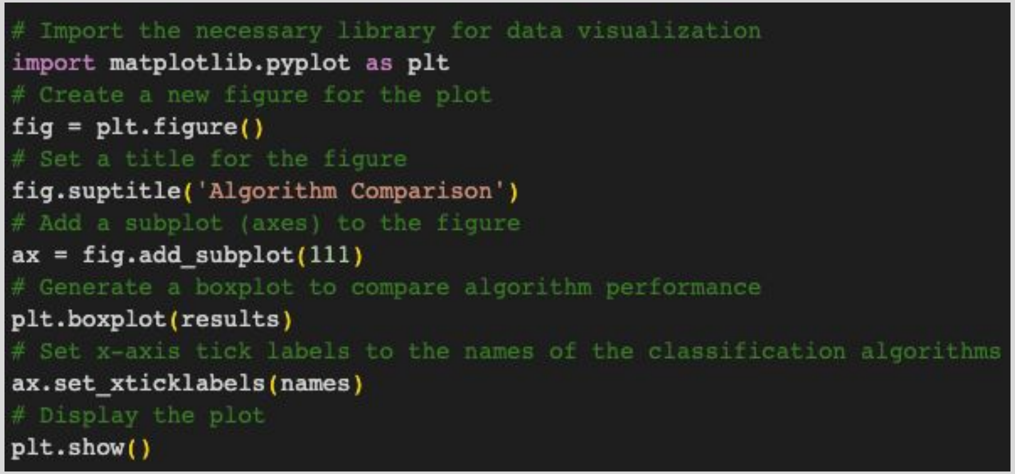
Performance

Compare the performance of different classification algorithms on the training set

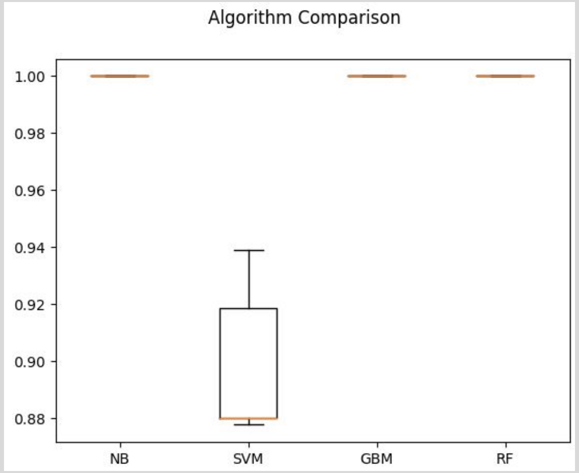
NB: Gaussian Naive Bayes
SVM: Support Vector Machine
GBM: Gradient Boosting
RF: Random Forest



```
Performance on Training set
NB: 1.000000 (0.000000)

SVM: 0.898939 (0.025057)

GBM: 1.000000 (0.000000)

RF: 1.000000 (0.000000)
```



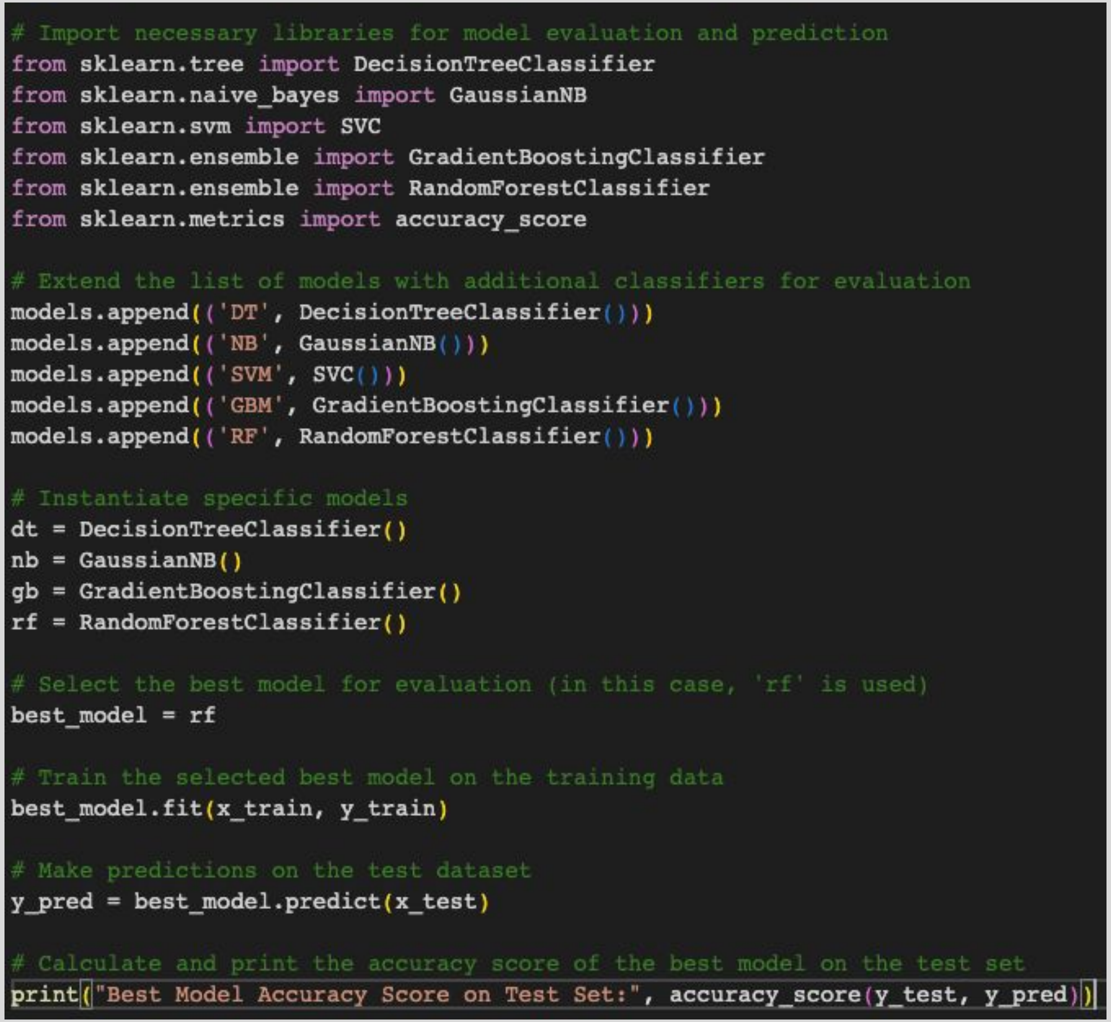
```
# Import the necessary library for data visualization
import matplotlib.pyplot as plt
# Create a new figure for the plot
fig = plt.figure()
# Set a title for the figure
fig.suptitle('Algorithm Comparison')
# Add a subplot (axes) to the figure
ax = fig.add_subplot(111)
# Generate a boxplot to compare algorithm performance
plt.boxplot(results)
# Set x-axis tick labels to the names of the classification algorithms
ax.set_xticklabels(names)
# Display the plot
plt.show()
```



2023

Visual Comparison

Provide a visual comparison of the performance of different machine learning algorithms on the training data

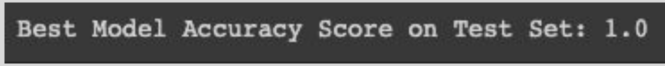


```
# Import necessary libraries for model evaluation and prediction
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Extend the list of models with additional classifiers for evaluation
models.append(('DT', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('GBM', GradientBoostingClassifier()))
models.append(('RF', RandomForestClassifier()))

# Instantiate specific models
dt = DecisionTreeClassifier()
nb = GaussianNB()
gb = GradientBoostingClassifier()
rf = RandomForestClassifier()

# Select the best model for evaluation (in this case, 'rf' is used)
best_model = rf

# Train the selected best model on the training data
best_model.fit(x_train, y_train)

# Make predictions on the test dataset
y_pred = best_model.predict(x_test)

# Calculate and print the accuracy score of the best model on the test set
print("Best Model Accuracy Score on Test Set:", accuracy_score(y_test, y_pred))
```

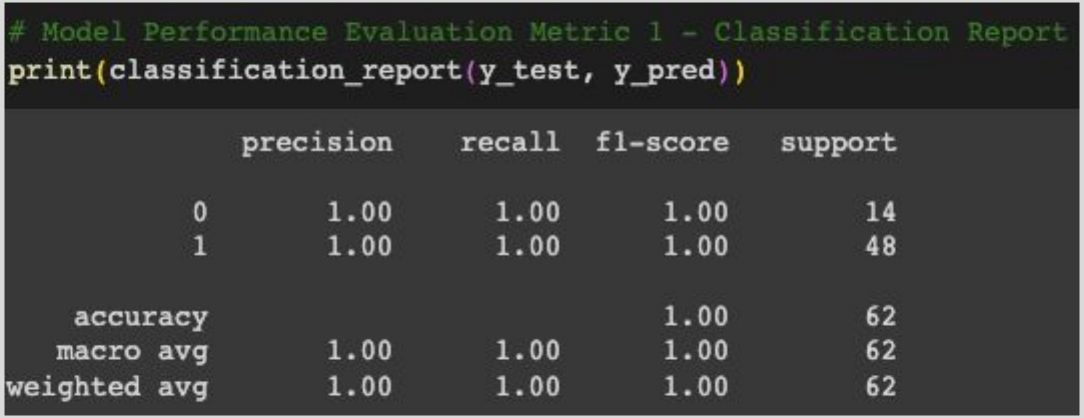
Best Model

Assess the performance of different classification models on an independent test dataset and determine which model achieves the highest accuracy on the test data.

NB: Gaussian Naive Bayes
SVM: Support Vector Machine
GBM: Gradient Boosting
RF: Random Forest



```
Best Model Accuracy Score on Test Set: 1.0
```



```
# Model Performance Evaluation Metric 1 - Classification Report
print(classification_report(y_test, y_pred))
```

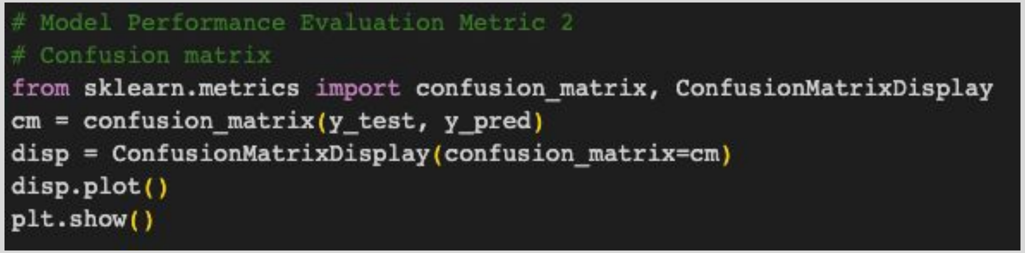
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        48

    accuracy                           1.00        62
   macro avg       1.00      1.00      1.00        62
weighted avg       1.00      1.00      1.00        62
```

2023

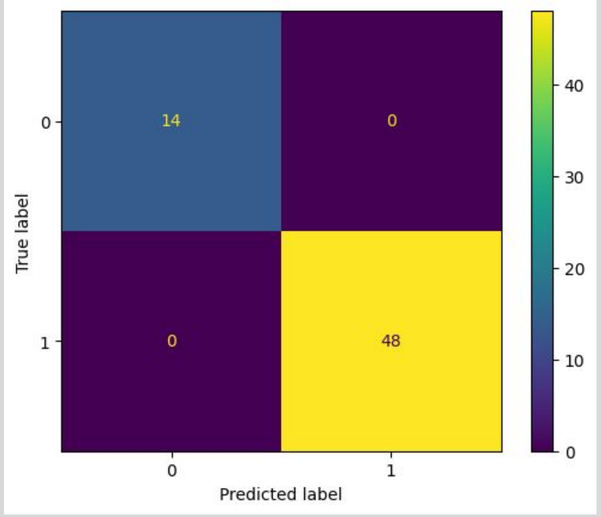
Classification

A detailed summary of the model's performance on the test dataset in a classification task

classification report is a valuable tool for understanding how well a model performs on individual classes within a multi-class classification problem



```
# Model Performance Evaluation Metric 2
# Confusion matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

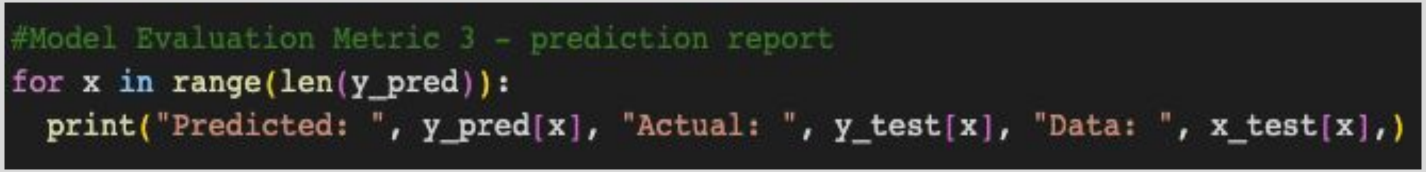


2023

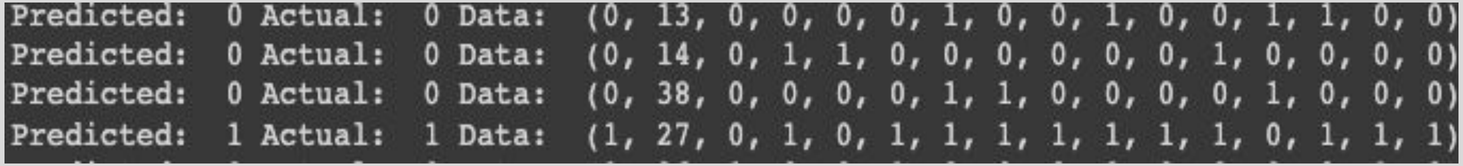
Confusion matrix

Visualize and evaluate the model's performance using a confusion matrix

Comprehensive view of how well the model performs in terms of correct and incorrect predictions for each class



```
#Model Evaluation Metric 3 - prediction report
for x in range(len(y_pred)):
  print("Predicted: ", y_pred[x], "Actual: ", y_test[x], "Data: ", x_test[x],)
```



```
Predicted:  0 Actual:  0 Data:  (0, 13, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0)
Predicted:  0 Actual:  0 Data:  (0, 14, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
Predicted:  0 Actual:  0 Data:  (0, 38, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0)
Predicted:  1 Actual:  1 Data:  (1, 27, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1)
```

2023

Prediction Report

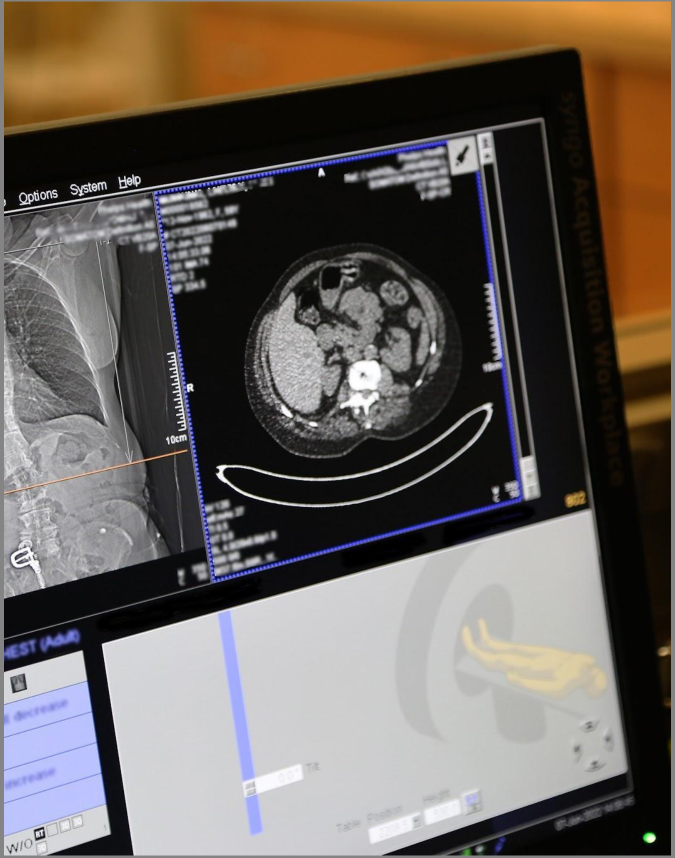
prediction report, where it displays the predicted values, actual values, and the corresponding data for each instance in the test dataset.

manual inspection of the model's predictions and how they compare to the actual values in the test datase

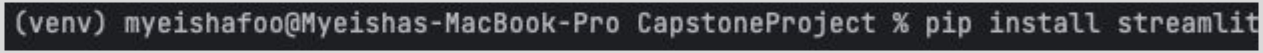
(03)

Deployment

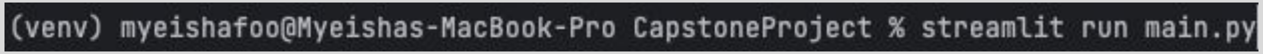
Implementation & Deployment



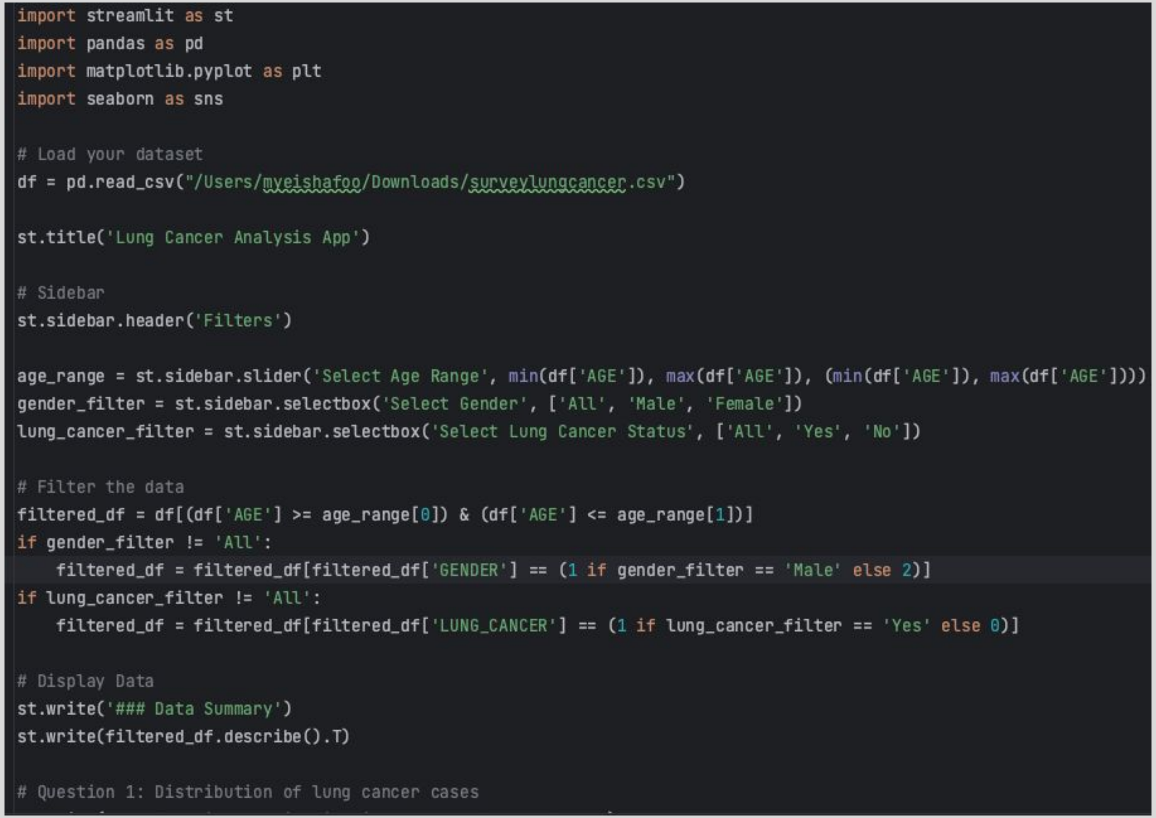
2022



```
(venv) myeishafoo@Myeishas-MacBook-Pro CapstoneProject % pip install streamlit
```



```
(venv) myeishafoo@Myeishas-MacBook-Pro CapstoneProject % streamlit run main.py
```



```python
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
df = pd.read_csv("/Users/myeishafoo/Downloads/surveylungcancer.csv")

st.title('Lung Cancer Analysis App')

# Sidebar
st.sidebar.header('Filters')

age_range = st.sidebar.slider('Select Age Range', min(df['AGE']), max(df['AGE']), (min(df['AGE']), max(df['AGE'])))
gender_filter = st.sidebar.selectbox('Select Gender', ['All', 'Male', 'Female'])
lung_cancer_filter = st.sidebar.selectbox('Select Lung Cancer Status', ['All', 'Yes', 'No'])

# Filter the data
filtered_df = df[(df['AGE'] >= age_range[0]) & (df['AGE'] <= age_range[1])]
if gender_filter != 'All':
    filtered_df = filtered_df[filtered_df['GENDER'] == (1 if gender_filter == 'Male' else 2)]
if lung_cancer_filter != 'All':
    filtered_df = filtered_df[filtered_df['LUNG_CANCER'] == (1 if lung_cancer_filter == 'Yes' else 0)]

# Display Data
st.write('### Data Summary')
st.write(filtered_df.describe().T)

# Question 1: Distribution of lung cancer cases
```

2023

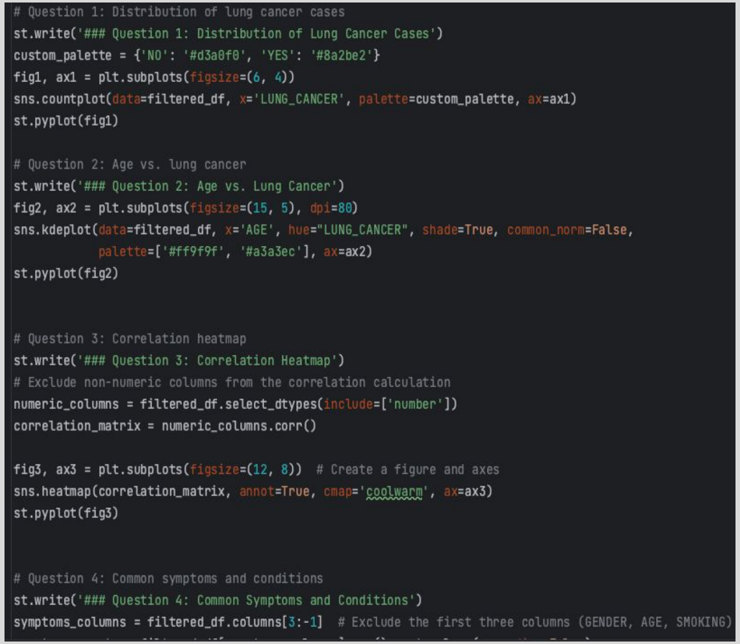
Install & Run

Streamlit web application that performs data analysis and visualization for lung cancer-related data

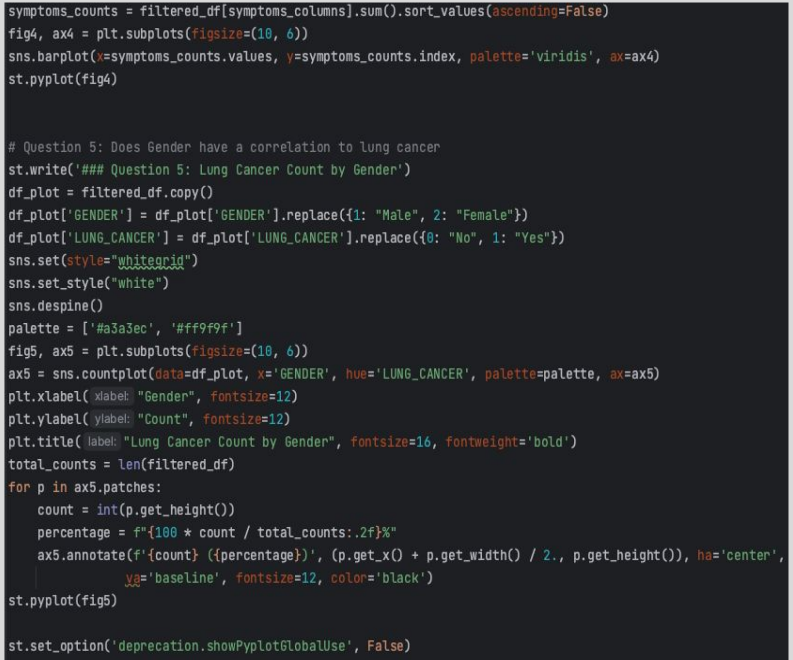
Questions

Create an interactive web application that allows users to explore and analyze lung cancer-related data with various visualization and data filtering options.

2023

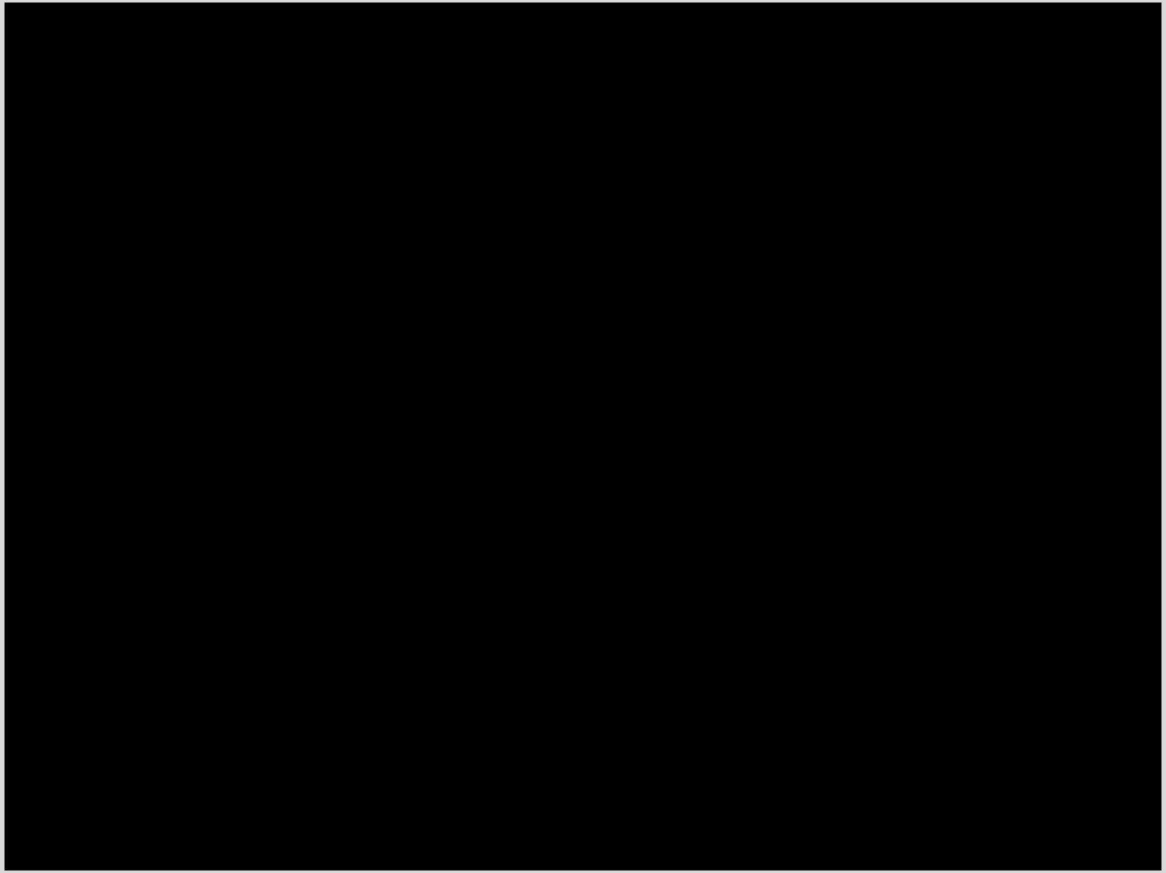


```python
# Question 1: Distribution of lung cancer cases
st.write('### Question 1: Distribution of Lung Cancer Cases')
custom_palette = {'NO': '#d3a0f0', 'YES': '#8a2be2'}
fig1, ax1 = plt.subplots(figsize=(6, 4))
sns.countplot(data=filtered_df, x='LUNG_CANCER', palette=custom_palette, ax=ax1)
st.pyplot(fig1)

# Question 2: Age vs. lung cancer
st.write('### Question 2: Age vs. Lung Cancer')
fig2, ax2 = plt.subplots(figsize=(15, 5), dpi=80)
sns.kdeplot(data=filtered_df, x='AGE', hue="LUNG_CANCER", shade=True, common_norm=False,
            palette=['#ff9f9f', '#a3a3ec'], ax=ax2)
st.pyplot(fig2)


# Question 3: Correlation heatmap
st.write('### Question 3: Correlation Heatmap')
# Exclude non-numeric columns from the correlation calculation
numeric_columns = filtered_df.select_dtypes(include=['number'])
correlation_matrix = numeric_columns.corr()

fig3, ax3 = plt.subplots(figsize=(12, 8))  # Create a figure and axes
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', ax=ax3)
st.pyplot(fig3)


# Question 4: Common symptoms and conditions
st.write('### Question 4: Common Symptoms and Conditions')
symptoms_columns = filtered_df.columns[3:-1]  # Exclude the first three columns (GENDER, AGE, SMOKING)
```



```python
symptoms_counts = filtered_df[symptoms_columns].sum().sort_values(ascending=False)
fig4, ax4 = plt.subplots(figsize=(10, 6))
sns.barplot(x=symptoms_counts.values, y=symptoms_counts.index, palette='viridis', ax=ax4)
st.pyplot(fig4)


# Question 5: Does Gender have a correlation to lung cancer
st.write('### Question 5: Lung Cancer Count by Gender')
df_plot = filtered_df.copy()
df_plot['GENDER'] = df_plot['GENDER'].replace({1: "Male", 2: "Female"})
df_plot['LUNG_CANCER'] = df_plot['LUNG_CANCER'].replace({0: "No", 1: "Yes"})
sns.set(style="whitegrid")
sns.set_style("white")
sns.despine()
palette = ['#a3a3ec', '#ff9f9f']
fig5, ax5 = plt.subplots(figsize=(10, 6))
ax5 = sns.countplot(data=df_plot, x='GENDER', hue='LUNG_CANCER', palette=palette, ax=ax5)
plt.xlabel( xlabel: "Gender", fontsize=12)
plt.ylabel( ylabel: "Count", fontsize=12)
plt.title( label: "Lung Cancer Count by Gender", fontsize=16, fontweight='bold')
total_counts = len(filtered_df)
for p in ax5.patches:
    count = int(p.get_height())
    percentage = f"{100 * count / total_counts:.2f}%"
    ax5.annotate(f'{count} ({percentage})', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center',
                 va='baseline', fontsize=12, color='black')
st.pyplot(fig5)

st.set_option('deprecation.showPyplotGlobalUse', False)
```

2023

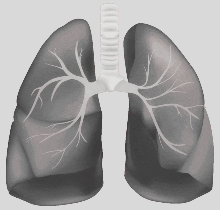


Streamlit Lung Cancer Analysis App

Streamlit web application that performs data analysis and visualization for lung cancer-related data

ST1(4483) Capstone Project

Thanks



Do you have any questions?

2023

References:

MySarahMadBhat. (2021). *Lung Cancer Dataset*. Kaggle. URL: https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer. Accessed: [12 October 2023]

Pandas Development Team. (2023). *Pandas DataFrame Documentation*. pandas.pydata.org. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html. Accessed: [1 October 2023]

Streamlit. (2023). *st.dataframe API Reference*. Streamlit Documentation. URL: https://docs.streamlit.io/library/api-reference/data/st.dataframe. Accessed: [2 October 2023].

Indeed. (2023). *How to Conduct Exploratory Data Analysis*. indeed.com. URL: https://www.indeed.com/career-advice/career-development/how-to-conduct-exploratory-data-analysis. Accessed: [31 September 2023]