FINAL PRESENTATION ON LUNG CANCER PREDICTION

Submitted by:

Bikesh Prajapati C0859472

Padam Regmi C0858265

Rosy Shrestha C0857467

Hemanta Rijal C0835075

Shreebatsa Aryal C0859473

LUNG CANCER PREDICTION USING DIFFERENT ML MODEL

IN THIS PROJECT, WE TRY TO PREDICT HIGH CHANCE TO GET CANCER AND ITS RISK FACTOR. SOLVE THE PROBLEM BY RECOMMENDING SOME SOLUTION AND BEST MODEL TO PREDICT WITH GIVEN DATA. WE DID SOME DATA VISUALIZATION TO LOOK INTO CORRELATION AND FEATURES AGAINST TARGET VARIABLES.

IMPORT THE REQUIRED LIBRARIES

```
import os
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, accuracy_score
from scipy.stats import shapiro
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from statsmodels.stats.diagnostic import normal_ad, het_breuschpagan
import statsmodels.api as sm
from sklearn.preprocessing import minmax_scale

%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv("cancer patient data sets.csv")

data.head()
```

DATA INFO

	index	Patient Id	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	...	Fatigue	Weight Loss	Shortness of Breath	Wheezing	Swallowing Difficulty	Clubbing of Finger Nails
0	0	P1	33	1	2	4	5	4	3	2	...	3	4	2	2	3	1
1	1	P10	17	1	3	1	5	3	4	2	...	1	3	7	8	6	2
2	2	P100	35	1	4	5	6	5	5	4	...	8	7	9	2	1	4
3	3	P1000	37	1	7	7	7	7	6	7	...	4	2	3	1	4	5
4	4	P101	46	1	6	8	7	7	7	6	...	3	2	4	1	4	2

5 rows × 26 columns

Cont …

Swallowing Difficulty	Clubbing of Finger Nails	Frequent Cold	Dry Cough	Snoring	Level
3	1	2	3	4	Low
6	2	1	7	2	Medium
1	4	6	7	2	High
4	5	6	7	5	High
4	2	4	2	3	High

- Importing the dataset using pd.read_csv() function
- **T** property is used to transpose index and columns of the data frame.
- The main function of this property is to create a reflection of the data frame overs the main diagonal by making rows as columns and vice versa.

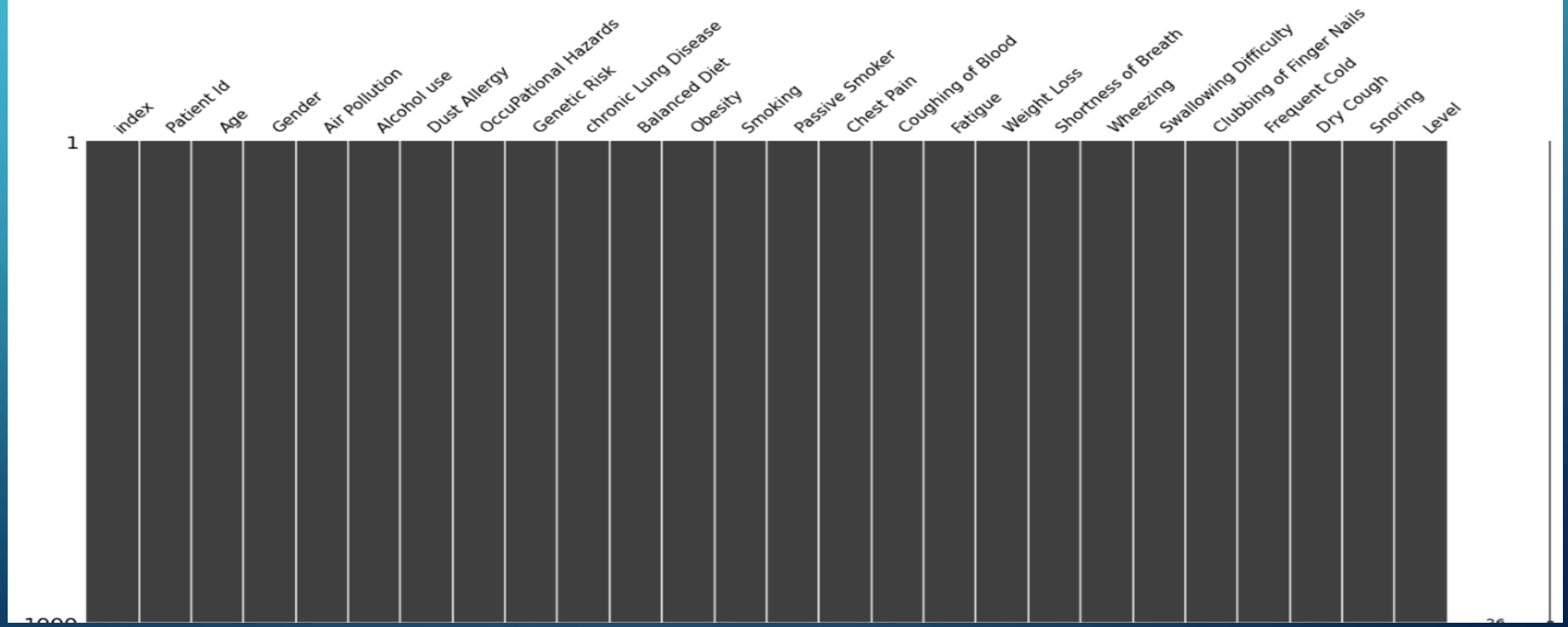
```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
index	1000.0	499.500	288.819436	0.0	249.75	499.5	749.25	999.0
Age	1000.0	37.174	12.005493	14.0	27.75	36.0	45.00	73.0
Gender	1000.0	1.402	0.490547	1.0	1.00	1.0	2.00	2.0
Air Pollution	1000.0	3.840	2.030400	1.0	2.00	3.0	6.00	8.0
Alcohol use	1000.0	4.563	2.620477	1.0	2.00	5.0	7.00	8.0
Dust Allergy	1000.0	5.165	1.980833	1.0	4.00	6.0	7.00	8.0
OccuPational Hazards	1000.0	4.840	2.107805	1.0	3.00	5.0	7.00	8.0
Genetic Risk	1000.0	4.580	2.126999	1.0	2.00	5.0	7.00	7.0
chronic Lung Disease	1000.0	4.380	1.848518	1.0	3.00	4.0	6.00	7.0
Balanced Diet	1000.0	4.491	2.135528	1.0	2.00	4.0	7.00	7.0
Obesity	1000.0	4.465	2.124921	1.0	3.00	4.0	7.00	7.0
Smoking	1000.0	3.948	2.495902	1.0	2.00	3.0	7.00	8.0
Passive Smoker	1000.0	4.195	2.311778	1.0	2.00	4.0	7.00	8.0
Chest Pain	1000.0	4.438	2.280209	1.0	2.00	4.0	7.00	9.0
Coughing of Blood	1000.0	4.859	2.427965	1.0	3.00	4.0	7.00	9.0

- Missingno is a Python library used for the visualization of missing data.
- We found out that the dataset has no missing value

```
import missingno as msno
msno.matrix(data)
```

```
<AxesSubplot:>
```



- The unique() function is used to get unique values of Series object and Returns the unique values as a NumPy array
- And here we got 'Low', 'Medium', 'High‘ as a unique value
- map() function is used to substitute each value in a Series with another value.
- We can see that the value of level column is 0,1 and 2 form which means low, medium and high

```
#checking the unique values of level and convert it into numerical values

data['Level'].unique()
```

```
array(['Low', 'Medium', 'High'], dtype=object)
```

Dataset Copy

```
data_copy = data.copy()
```

Covert level into numerical represntative

```
data_copy['Level'] = data_copy['Level'].map({'Low':0,'Medium':1,'High':2})
```

```
data_copy.head()
```

	index	Patient Id	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	...	Fatigue	Weight Loss	Shortness of Breath	Wheezing	Swallowing Difficulty	Clubbing of Finger Nails
0	0	P1	33	1	2	4	5	4	3	2	...	3	4	2	2	3	1
1	1	P10	17	1	3	1	5	3	4	2	...	1	3	7	8	6	2
2	2	P100	35	1	4	5	6	5	5	4	...	8	7	9	2	1	4
3	3	P1000	37	1	7	7	7	7	6	7	...	4	2	3	1	4	5
4	4	P101	46	1	6	8	7	7	7	6	...	3	2	4	1	4	2

- **Drop user's column**

Index and Patient Id does not affect our prediction of lung cancer so we can drop those columns

Index and Patient Id does not affect out prediction of lung cancer so we can drop those columns

```
data_copy = data_copy.drop(['index','Patient Id'], axis = 1)
```

- **Data Cleaning**

Data cleansing identifies and fixes errors, duplicates, and irrelevant data from a raw dataset allows for accurate, defensible data that generates reliable visualizations, models, and business decisions.

```
def duplicates(df):
    """
    Remove the duplicate rows from dataframe.

    Parameters
    ----------
    df: Pandas dataframe

    Returns
    -------
    df: Pandas dataframe without duplicate rows
    """
    duplicate_rows_df = df[df.duplicated()]
    if duplicate_rows_df.shape[0] > 0:
       print('Number of rows before removing:', df.count()[0])
       print('Number of duplicate rows:', duplicate_rows_df.shape[0])
       df = df.drop_duplicates()
       print('Number of rows after removing:', df.count()[0])
    else:
       print('No duplicate rows.')
    return df
```

- To see the **duplicate Series values**
- We used the duplicated() function

```
data_copy = duplicates(data_copy)
```

```
Number of rows before removing: 1000
Number of duplicate rows: 848
Number of rows after removing: 152
```

DATA VISUALIZATION TO SEE CORRELATION

- We used the heatmap to see the relationships between two variables one plotted on each axis
- According to heat map , the target variable 'Level' is highly correlated to following columns:

Air pollution

Alcohol use

Dust Allergy

Occupational Hazards

Genetic Risk

Chronic Lung Disease

Balance Diet

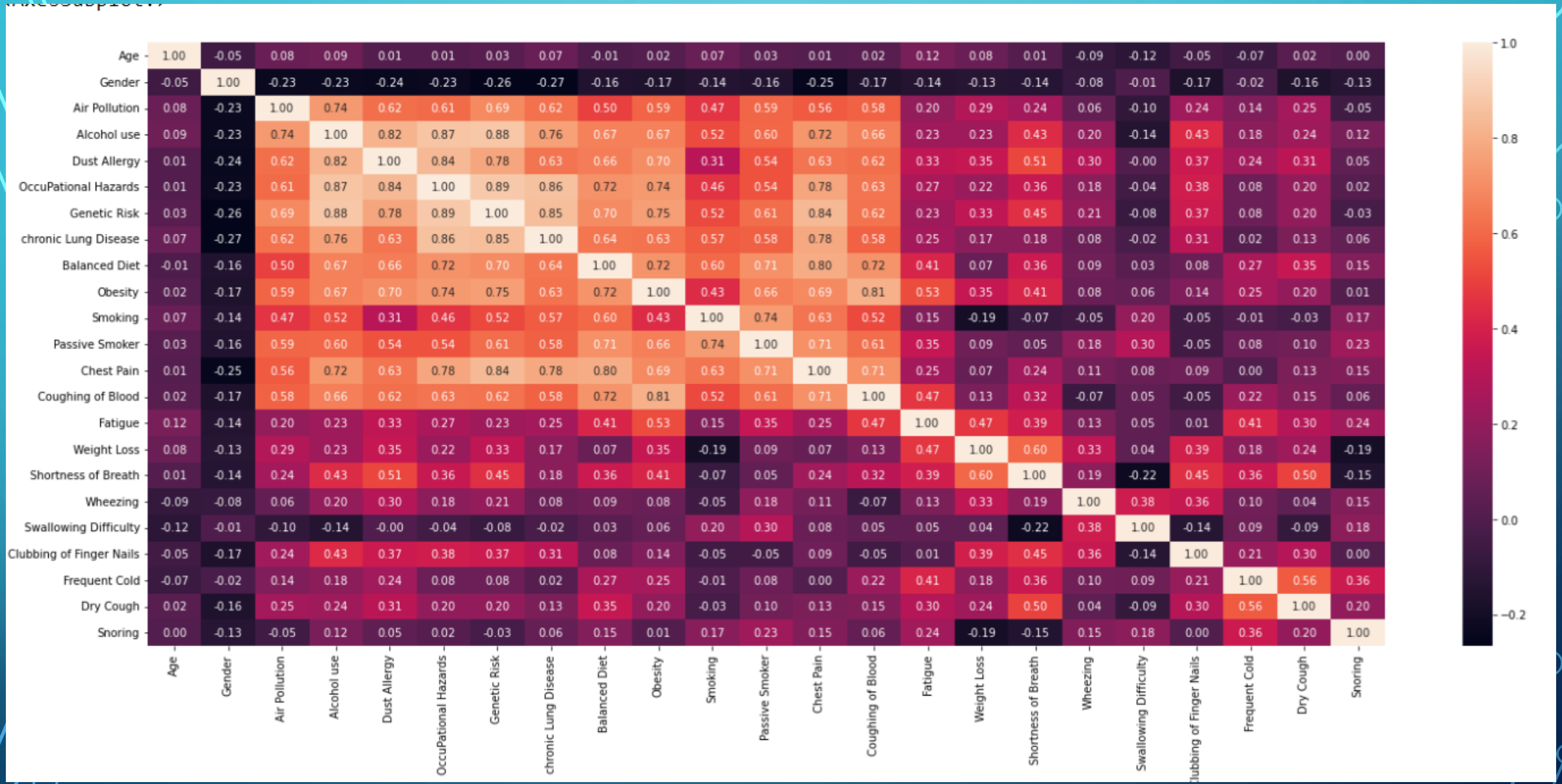
Obesity

Smoking

Passive Smoker

Chest Pain

Coughing of Blood

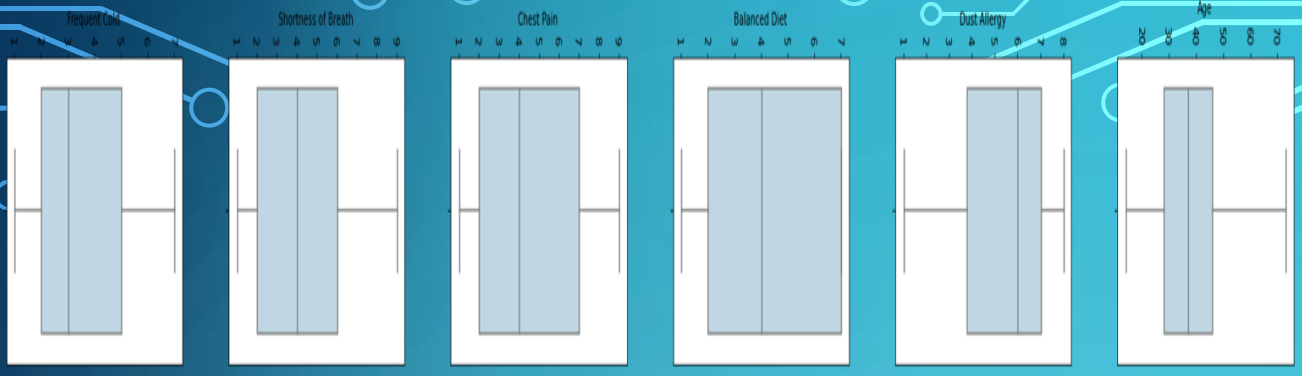

Age 1.00 -0.05 0.08 0.09 0.01 0.01 0.03 0.07 -0.01 0.02 0.07 0.03 0.01 0.02 0.12 0.08 0.01 -0.09 -0.12 -0.05 -0.07 0.02 0.00
Gender -0.05 1.00 -0.23 -0.23 -0.24 -0.23 -0.26 -0.27 -0.16 -0.17 -0.14 -0.16 -0.25 -0.17 -0.14 -0.13 -0.14 -0.08 -0.01 -0.17 -0.02 -0.16 -0.13
Air Pollution 0.08 -0.23 1.00 0.74 0.62 0.61 0.69 0.62 0.50 0.59 0.47 0.59 0.56 0.58 0.20 0.29 0.24 0.06 -0.10 0.24 0.14 0.25 -0.05
Alcohol use 0.09 -0.23 0.74 1.00 0.82 0.87 0.88 0.76 0.67 0.67 0.52 0.60 0.72 0.66 0.23 0.23 0.43 0.20 -0.14 0.43 0.18 0.24 0.12
Dust Allergy 0.01 -0.24 0.62 0.82 1.00 0.84 0.78 0.63 0.66 0.70 0.31 0.54 0.63 0.62 0.33 0.35 0.51 0.30 -0.00 0.37 0.24 0.31 0.05
OccuPational Hazards 0.01 -0.23 0.61 0.87 0.84 1.00 0.89 0.86 0.72 0.74 0.46 0.54 0.78 0.63 0.27 0.22 0.36 0.18 -0.04 0.38 0.08 0.20 0.02
Genetic Risk 0.03 -0.26 0.69 0.88 0.78 0.89 1.00 0.85 0.70 0.75 0.52 0.61 0.84 0.62 0.23 0.33 0.45 0.21 -0.08 0.37 0.08 0.20 -0.03
chronic Lung Disease 0.07 -0.27 0.62 0.76 0.63 0.86 0.85 1.00 0.64 0.63 0.57 0.58 0.78 0.58 0.25 0.17 0.18 0.08 -0.02 0.31 0.02 0.13 0.06
Balanced Diet -0.01 -0.16 0.50 0.67 0.66 0.72 0.70 0.64 1.00 0.72 0.60 0.71 0.80 0.72 0.41 0.07 0.36 0.09 0.03 0.08 0.27 0.35 0.15
Obesity 0.02 -0.17 0.59 0.67 0.70 0.74 0.75 0.63 0.72 1.00 0.43 0.66 0.69 0.81 0.53 0.35 0.41 0.08 0.06 0.14 0.25 0.20 0.01
Smoking 0.07 -0.14 0.47 0.52 0.31 0.46 0.52 0.57 0.60 0.43 1.00 0.74 0.63 0.52 0.15 -0.19 -0.07 -0.05 0.20 -0.05 -0.01 -0.03 0.17
Passive Smoker 0.03 -0.16 0.59 0.60 0.54 0.54 0.61 0.58 0.71 0.66 0.74 1.00 0.71 0.61 0.35 0.09 0.05 0.18 0.30 -0.05 0.08 0.10 0.23
Chest Pain 0.01 -0.25 0.56 0.72 0.63 0.78 0.84 0.78 0.80 0.69 0.63 0.71 1.00 0.71 0.25 0.07 0.24 0.11 0.08 0.09 0.00 0.13 0.15
Coughing of Blood 0.02 -0.17 0.58 0.66 0.62 0.63 0.62 0.58 0.72 0.81 0.52 0.61 0.71 1.00 0.47 0.13 0.32 -0.07 0.05 -0.05 0.22 0.15 0.06
Fatigue 0.12 -0.14 0.20 0.23 0.33 0.27 0.23 0.25 0.41 0.53 0.15 0.35 0.25 0.47 1.00 0.47 0.39 0.13 0.05 0.01 0.41 0.30 0.24
Weight Loss 0.08 -0.13 0.29 0.23 0.35 0.22 0.33 0.17 0.07 0.35 -0.19 0.09 0.07 0.13 0.47 1.00 0.60 0.33 0.04 0.39 0.18 0.24 -0.19
Shortness of Breath 0.01 -0.14 0.24 0.43 0.51 0.36 0.45 0.18 0.36 0.41 -0.07 0.05 0.24 0.32 0.39 0.60 1.00 0.19 -0.22 0.45 0.36 0.50 -0.15
Wheezing -0.09 -0.08 0.06 0.20 0.30 0.18 0.21 0.08 0.09 0.08 -0.05 0.18 0.11 -0.07 0.13 0.33 0.19 1.00 0.38 0.36 0.10 0.04 0.15
Swallowing Difficulty -0.12 -0.01 -0.10 -0.14 -0.00 -0.04 -0.08 -0.02 0.03 0.06 0.20 0.30 0.08 0.05 0.05 0.04 -0.22 0.38 1.00 -0.14 0.09 -0.09 0.18
Clubbing of Finger Nails -0.05 -0.17 0.24 0.43 0.37 0.38 0.37 0.31 0.08 0.14 -0.05 -0.05 0.09 -0.05 0.01 0.39 0.45 0.36 -0.14 1.00 0.21 0.30 0.00
Frequent Cold -0.07 -0.02 0.14 0.18 0.24 0.08 0.08 0.02 0.27 0.25 -0.01 0.08 0.00 0.22 0.41 0.18 0.36 0.10 0.09 0.21 1.00 0.56 0.36
Dry Cough 0.02 -0.16 0.25 0.24 0.31 0.20 0.20 0.13 0.35 0.20 -0.03 0.10 0.13 0.15 0.30 0.24 0.50 0.04 -0.09 0.30 0.56 1.00 0.20
Snoring 0.00 -0.13 -0.05 0.12 0.05 0.02 -0.03 0.06 0.15 0.01 0.17 0.23 0.15 0.06 0.24 -0.19 -0.15 0.15 0.18 0.00 0.36 0.20 1.00
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
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

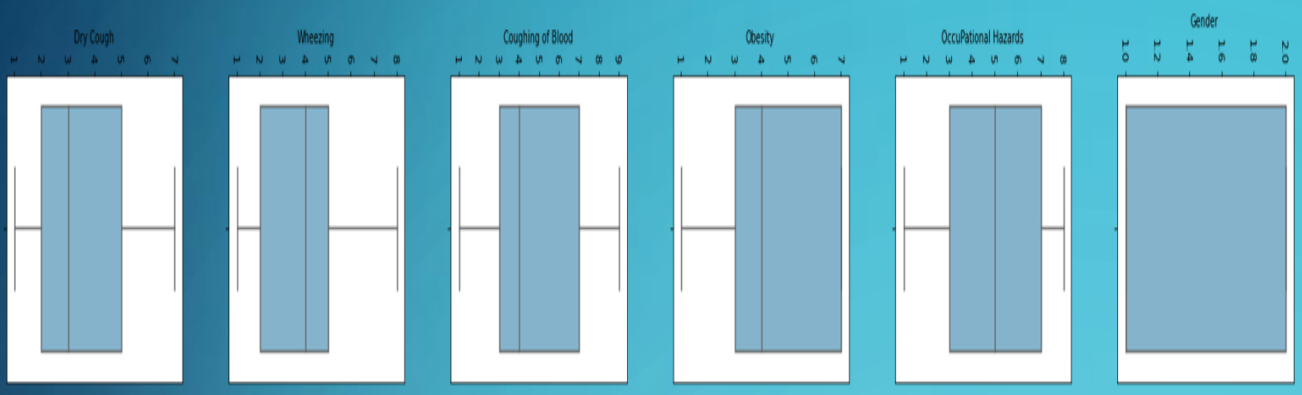
CHECKING IF THERE IS ANY OUTLIER

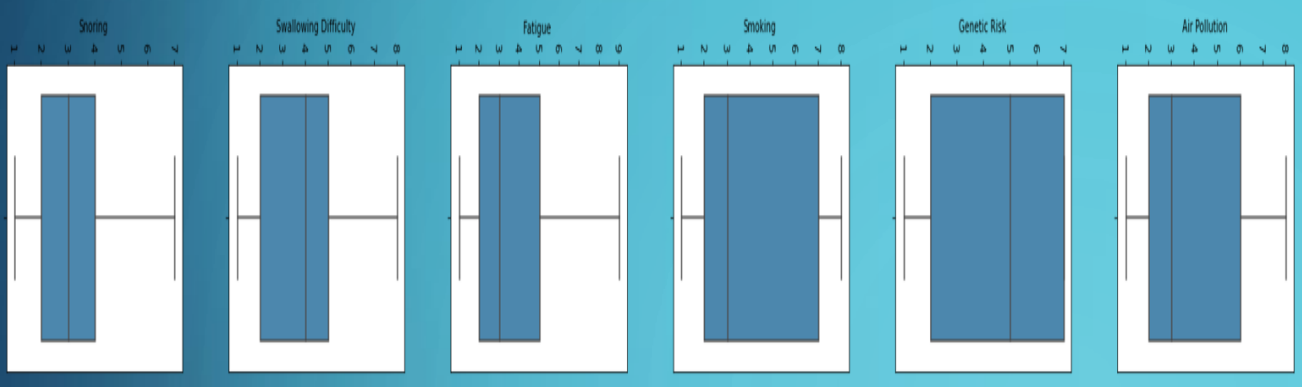
- We used boxplot to identify the outlier, we can see the data point that is located outside the whiskers of the box plot
- we analyzed that only Age has some outlier in the data set.

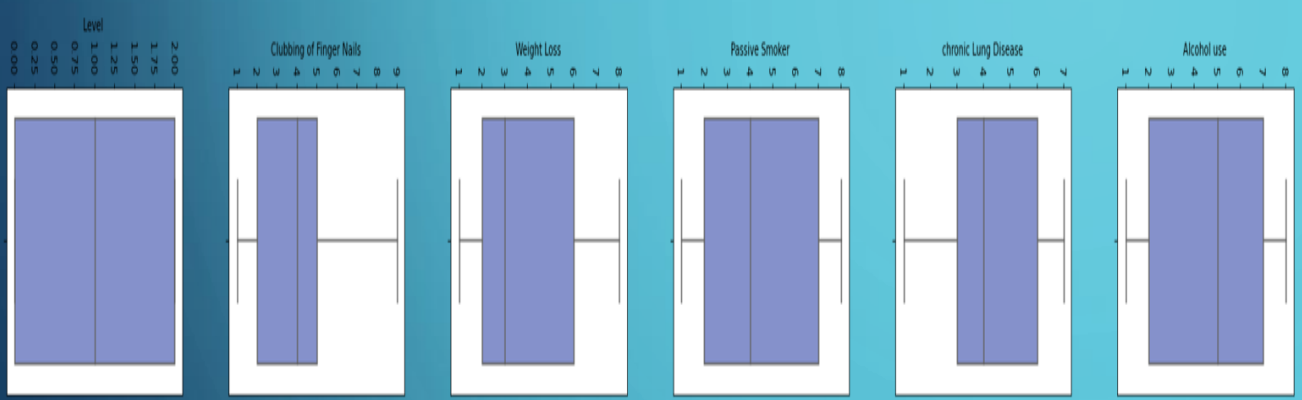
checking if there is any outlier

```
def plotBoxplot(data):
    fig, axes = plt.subplots(ncols=4, nrows=6, figsize=(15,25))
    fig.tight_layout(pad=4.0)

    col = 0
    row = 0
    colors = ['#bad9e9', '#7ab6d6', '#3c8abd','#7a89d6']

    for i, column in enumerate(data.columns):
        sns.boxplot(y=column, data=data, ax=axes[row][col], color=colors[col])

        if (i + 1) % 4 == 0:
            row += 1
            col = 0
        else:
            col += 1

plotBoxplot(data_copy)
```

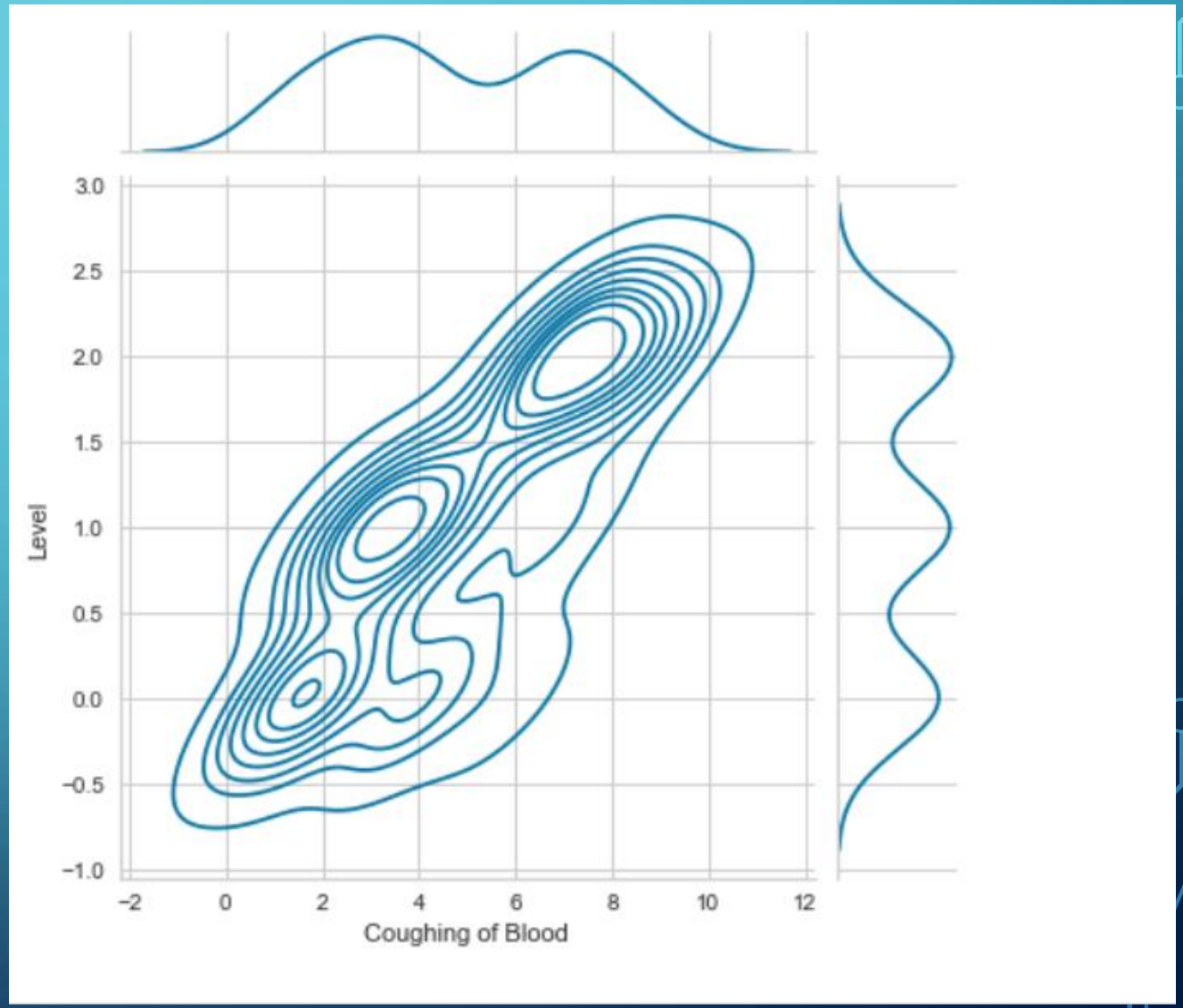

Frequent Cold
Shortness of Breath
Chest Pain
Balanced Diet
Dust Allergy
Age


Dry Cough
Wheezing
Coughing of Blood
Obesity
OccuPational Hazards
Gender

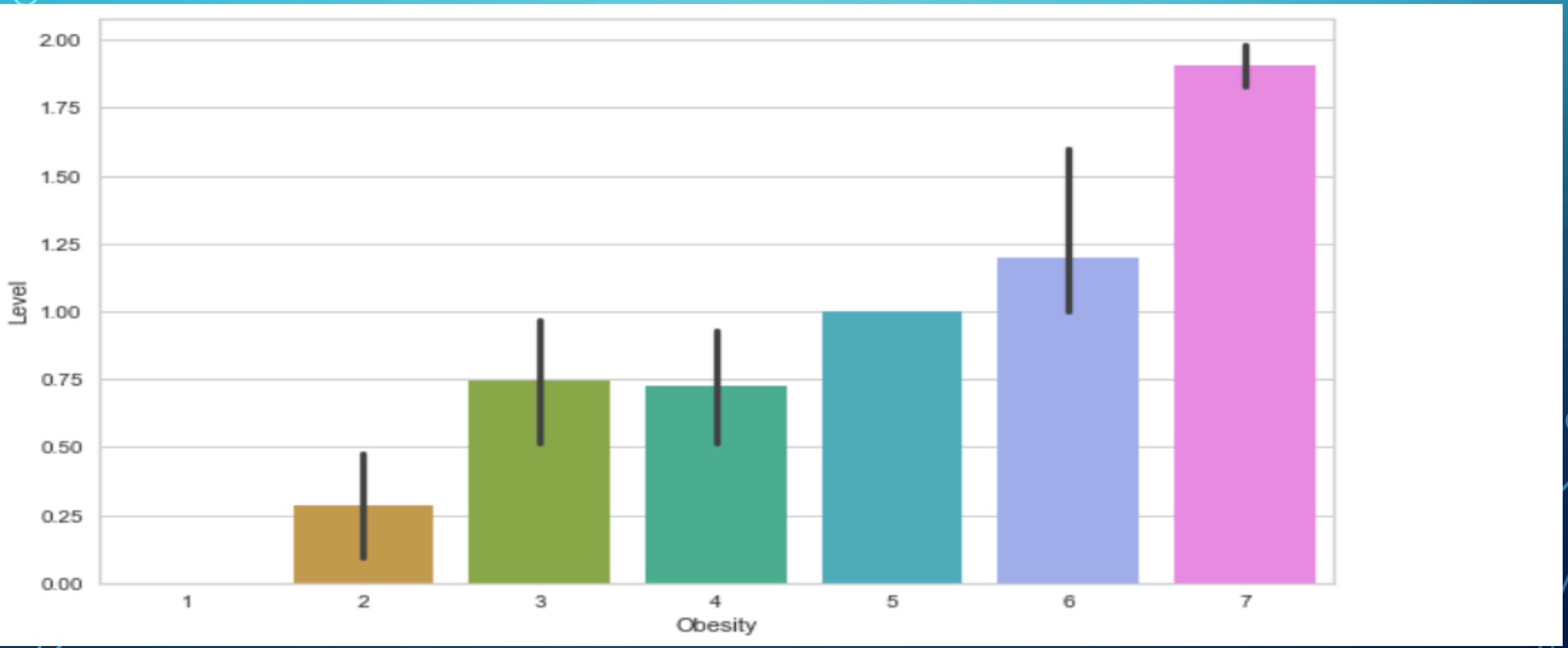

Snoring
Swallowing Difficulty
Fatigue
Smoking
Genetic Risk
Air Pollution


Level
Clubbing of Finger Nails
Weight Loss
Passive Smoker
chronic Lung Disease
Alcohol use

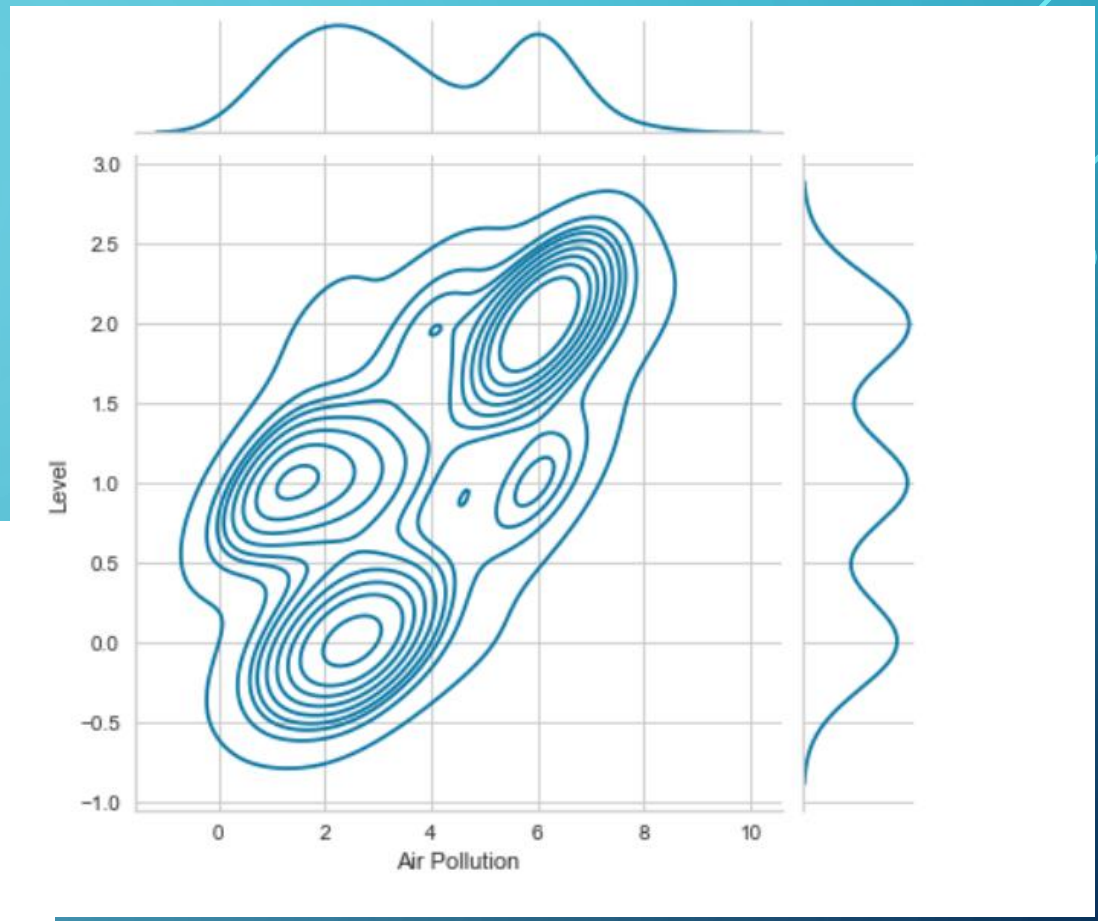
- Analyzing Level with coughing blood using kernel density estimation plot.
- KDE is a useful technique to create a smooth curve given a set of data.
- We have plotted the bar-graph that shows that higher the level of coughing blood higher will be the chance of cancer.

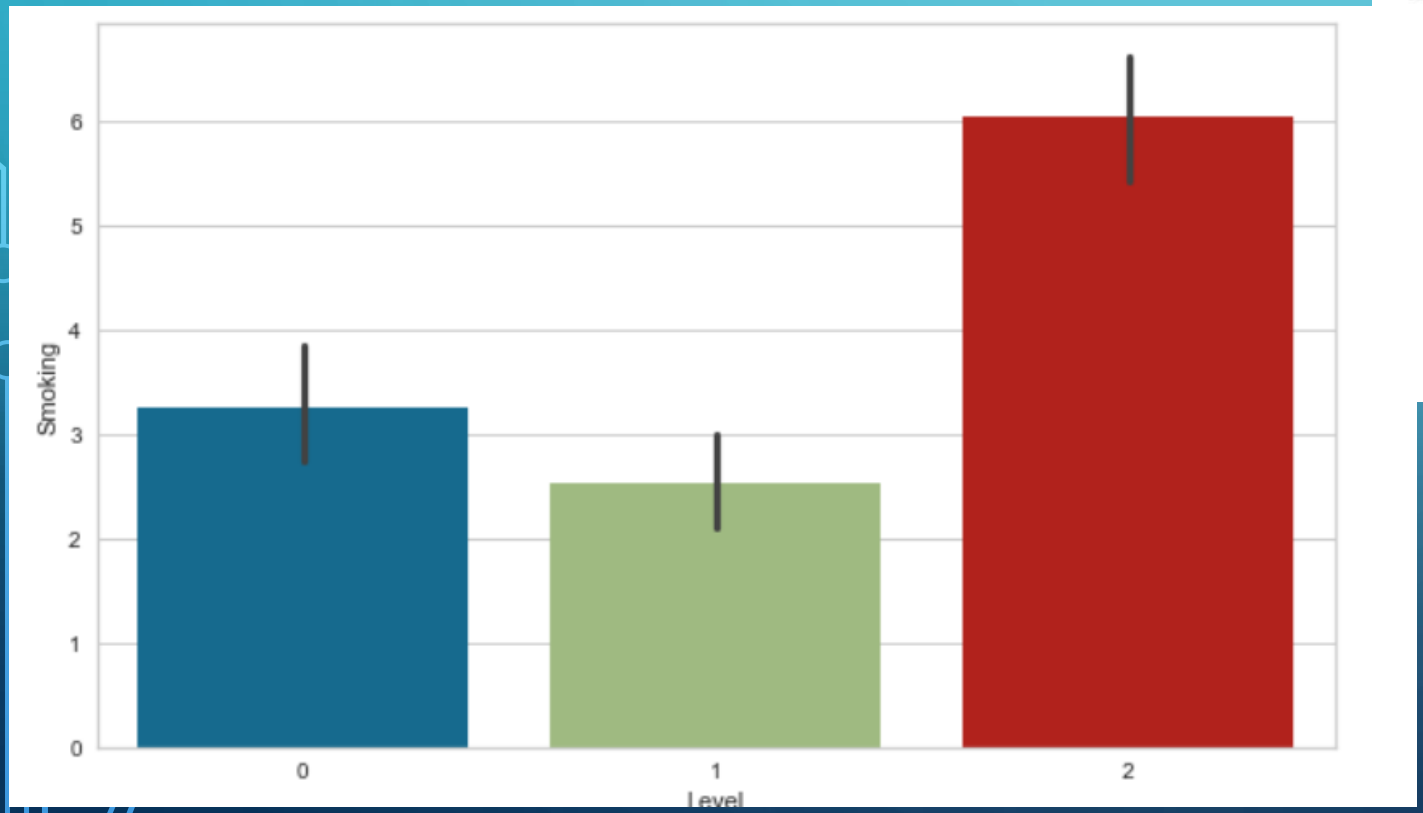


- We also plotted the bar plot to see the **level over obesity** and we can see that it's quite a upward trend in the obesity as we go higher. Higher the obesity higher will be the chance of cancer.

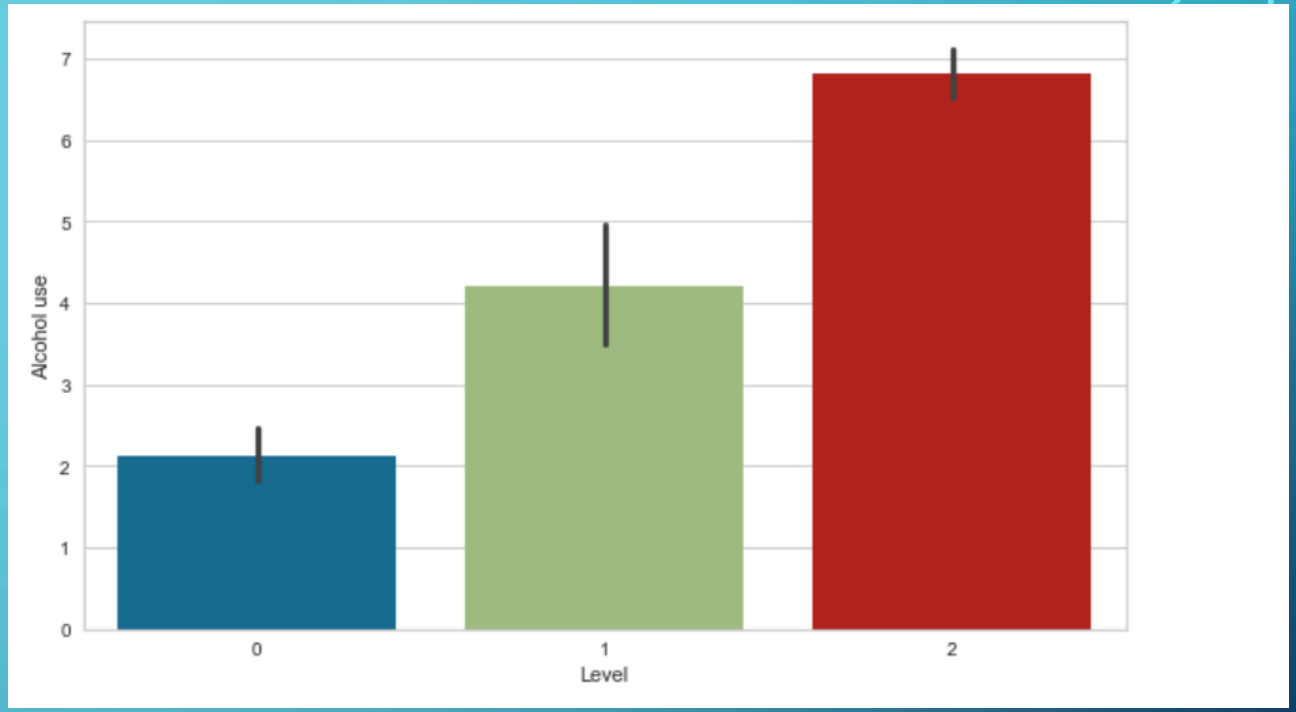


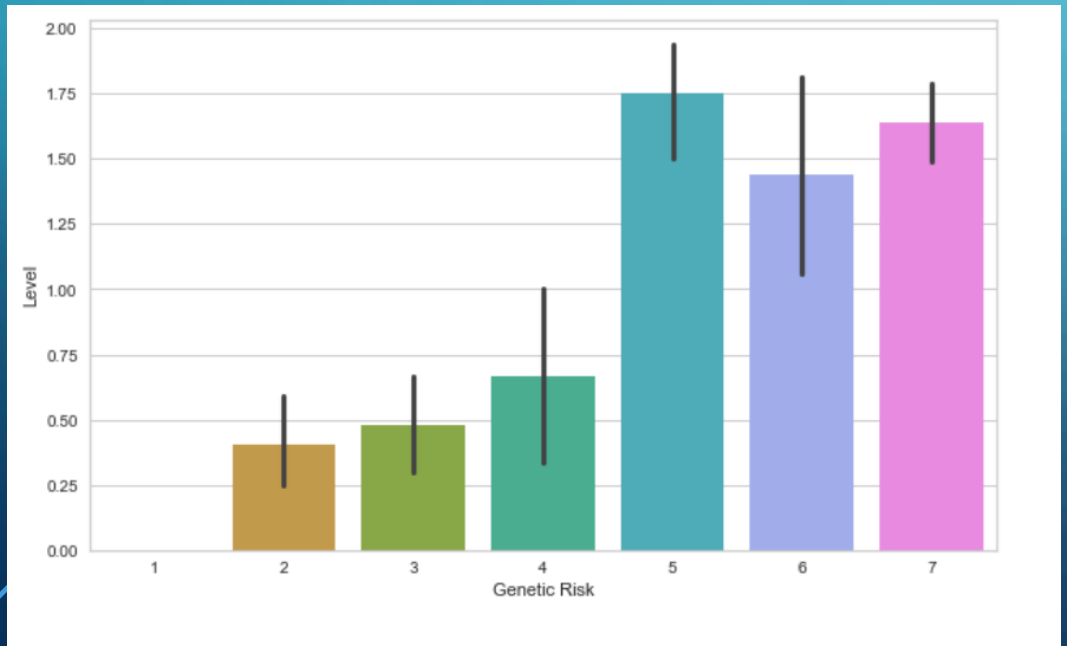
- We have plotted the joint plot for **level over Air-pollution**
- Also we have also plotted the bar plot for **level over Smoking** and we can see that it's quite a upward trend in the smoking as we go higher the smoking level we go higher level chance of cancer





- We have also plotted bar graph for **Alcohol use over Level** and we can see that there is upward trend in the Level as we go higher use of alcohol level we see higher chance of cancer.
- Also for the bar-plot **for Genetic Risk over level** and we can observe that the people having genetic risk has the high chance of getting cancer.



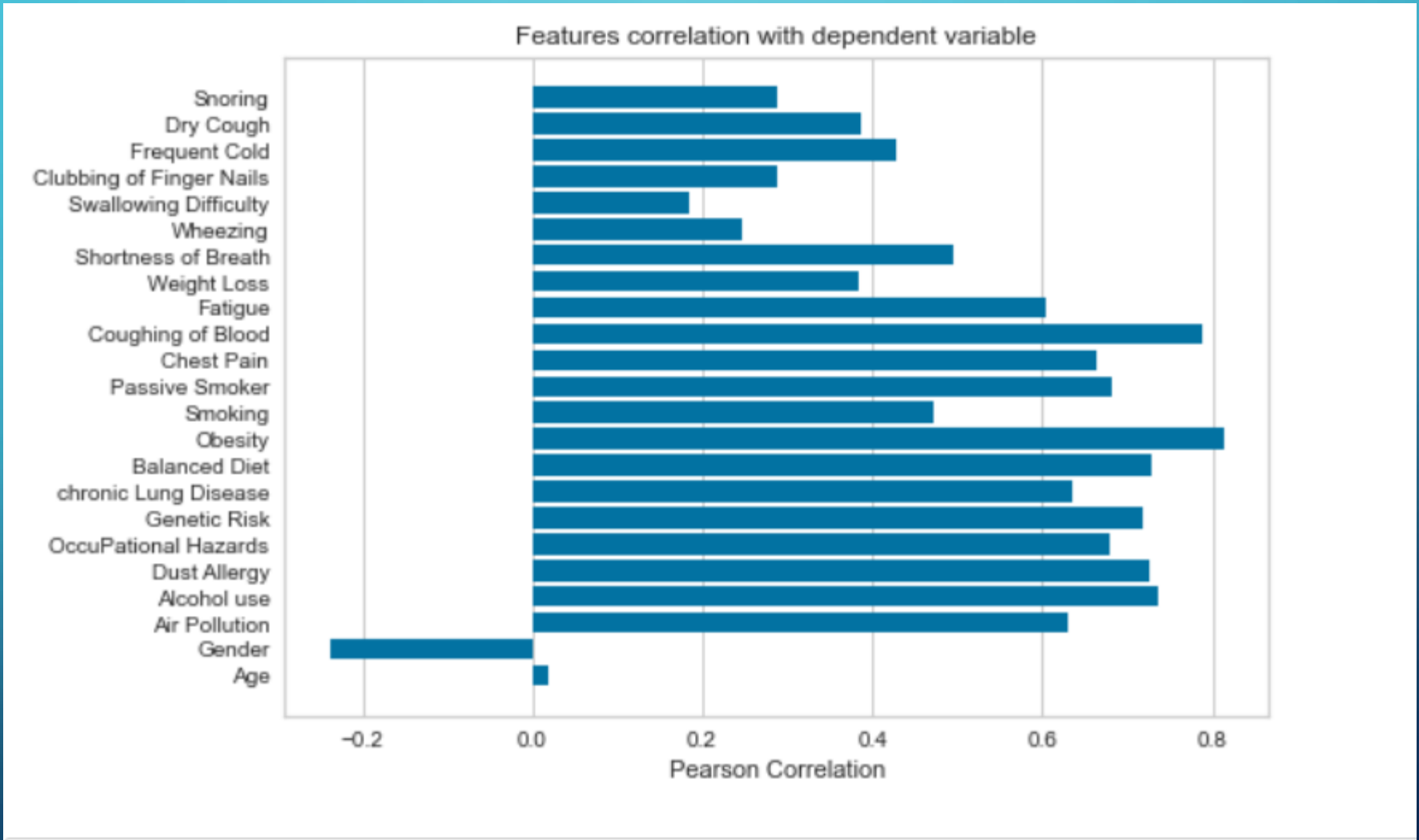


SPLITTING DATA-FRAME IN FEATURES AND TARGETS

- We have splitted the dataset into **training and testing sets** where for training the model we have taken all the features except level and for testing we have taken the level column as input.
- After that we did the visualization using feature correlation library which is used in feature selection to identify features with high correlation or large mutual information with the dependent variable.
- Train Test Ratio is 80:20

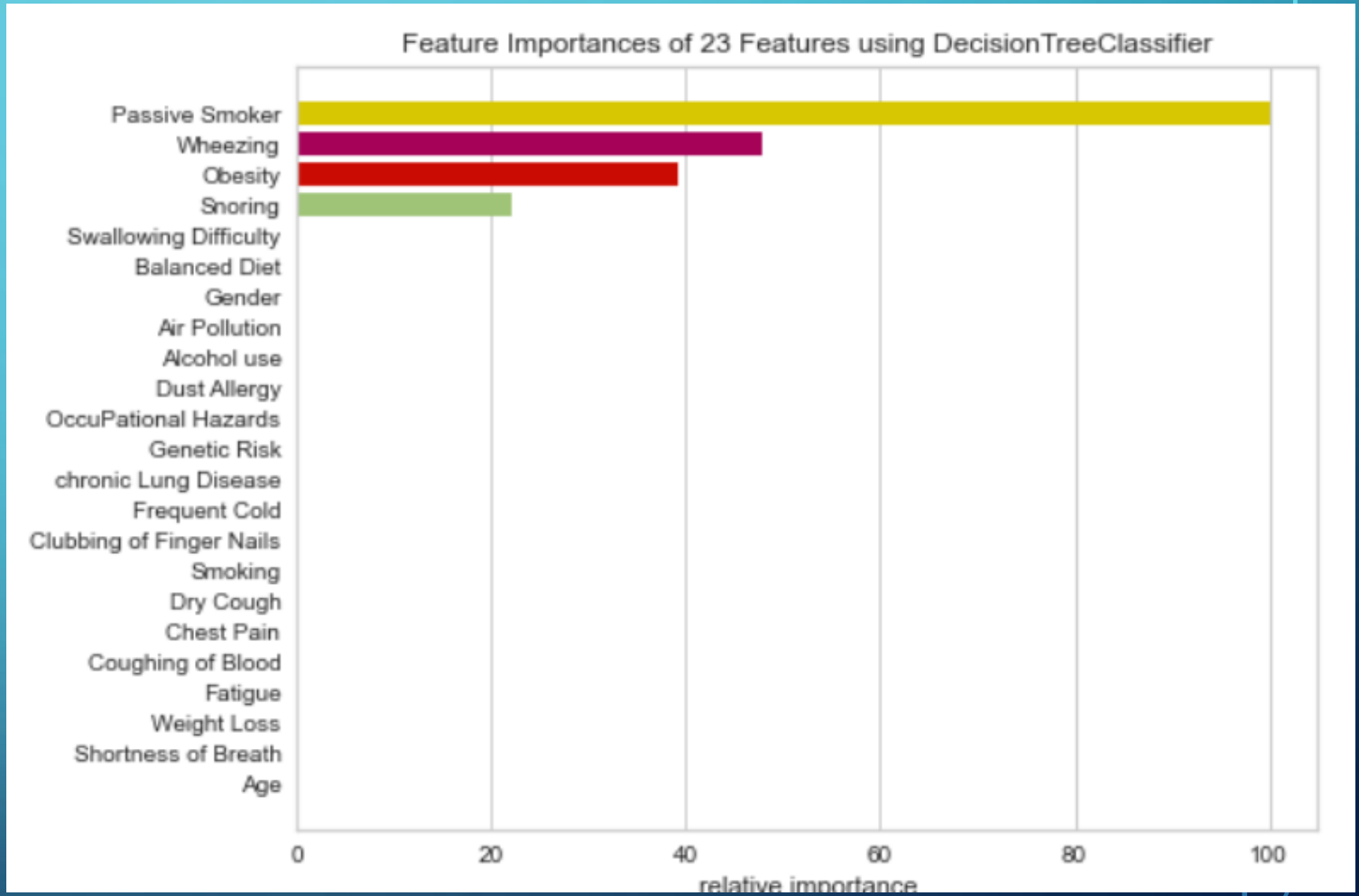
Spliting dataFrame in features and targets

```
#create tmp train/test split for assumptions test
X = data_copy.drop(['Level'], axis=1)
y = data_copy['Level']
```


Features correlation with dependent variable
Snoring
Dry Cough
Frequent Cold
Clubbing of Finger Nails
Swallowing Difficulty
Wheezing
Shortness of Breath
Weight Loss
Fatigue
Coughing of Blood
Chest Pain
Passive Smoker
Smoking
Obesity
Balanced Diet
chronic Lung Disease
Genetic Risk
OccuPational Hazards
Dust Allergy
Alcohol use
Air Pollution
Gender
Age
−0.2
0.0
0.2
0.4
0.6
0.8
Pearson Correlation

FEATURE IMPORTANCE

- calculate a score for all the input features.
- According to the feature we can conclude that passive smoker, wheezing, obesity and snoring is highly relative importance.



PCA IMPLEMENTATION

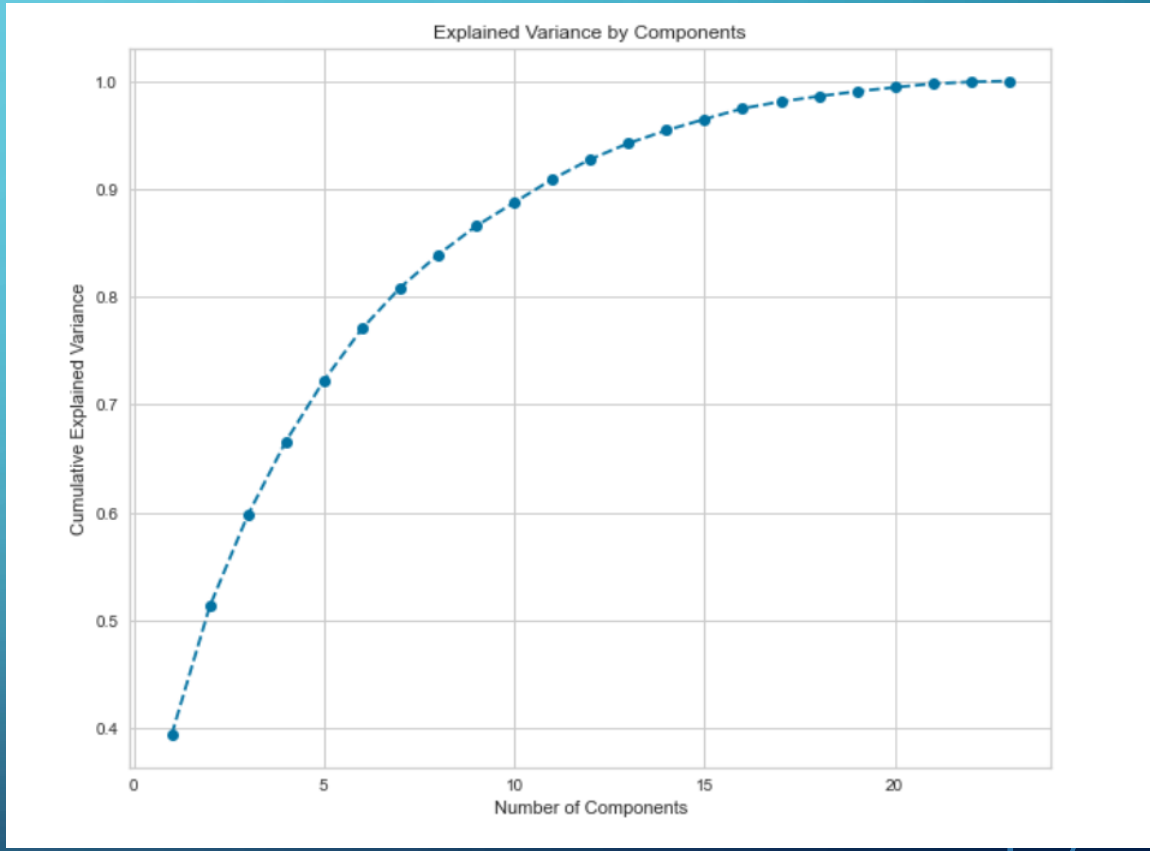
```
# we use standard scaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(X)
```

```
PCA()
```

```
pca.explained_variance_ratio_
```

```
array([0.39443955, 0.1187636 , 0.08462439, 0.06827646, 0.05640203,
       0.04799772, 0.03791487, 0.03063632, 0.0265428 , 0.0218407 ,
       0.02158494, 0.01852366, 0.01475467, 0.01199977, 0.01032852,
       0.00996056, 0.00628791, 0.00494126, 0.00450478, 0.00380493,
       0.00338612, 0.00162582, 0.00085862])
```



TEST TRAIN SPLIT

Splitting data in train and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2 ,random_state=50)
```

Build Model

Table of models and their scores

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model", "Train Score", "Test Score", "R2 Score", "Mean Absolute Error", "Mean Squared Error"]
```

Model We are Using

1. Linear Regression
2. Lasso Regression
3. Random Forest Regression
4. Gradient Boosting Regression.

Splitting data in train and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2 ,random_state=50)
```

Build Model

Table of models and their scores

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model", "Train Score", "Test Score", "R2 Score", "Mean Absolute Error", "Mean Squared Error"]
```

LINEAR REGRESSION

Model 1: Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_train_score = lr.score(X_train, y_train)
lr_test_score = lr.score(X_test, y_test)
lr_r2_score = r2_score(y_test, lr.predict(X_test))
lr_mae = mean_absolute_error(y_test, lr.predict(X_test))
lr_mse = mean_squared_error(y_test, lr.predict(X_test))
x.add_row(["Linear Regression", lr_train_score, lr_test_score, lr_r2_score, lr_mae, lr_mse])

print(x)
```

LASSO REGRESSION

Model 2: Lasso Regression

```
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
ls = Lasso(alpha=0.001)
ls.fit(X_train, y_train)
ls_train_score = ls.score(X_train, y_train)
ls_test_score = ls.score(X_test, y_test)
ls_r2_score = r2_score(y_test, ls.predict(X_test))
ls_mae = mean_absolute_error(y_test, ls.predict(X_test))
ls_mse = mean_squared_error(y_test, ls.predict(X_test))
x.add_row(["Lasso Regression", ls_train_score, ls_test_score, ls_r2_score, ls_mae, ls_mse])

print(x)
```

RANDOM FOREST REGRESSION

Model 3: Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor()
rfr.fit(X_train, y_train)
rfr_train_score = rfr.score(X_train, y_train)
rfr_test_score = rfr.score(X_test, y_test)
rfr_r2_score = r2_score(y_test, rfr.predict(X_test))
rfr_mae = mean_absolute_error(y_test, rfr.predict(X_test))
rfr_mse = mean_squared_error(y_test, rfr.predict(X_test))
x.add_row(["Random Forest Regressor", rfr_train_score, rfr_test_score, rfr_r2_score, rfr_mae, rfr_mse])

print(x)
```

GRADIENT BOOSTING REGRESSOR

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)
gbr_train_score = gbr.score(X_train, y_train)
gbr_test_score = gbr.score(X_test, y_test)
gbr_r2_score = r2_score(y_test, gbr.predict(X_test))
gbr_mae = mean_absolute_error(y_test, gbr.predict(X_test))
gbr_mse = mean_squared_error(y_test, gbr.predict(X_test))
x.add_row(["Gradient Boosting Regressor", gbr_train_score, gbr_test_score, gbr_r2_score, gbr_mae, gbr_mse])

print(x)
```

Comparission between different models

```
print(x)
```

```
+-----------------------------+--------------------+--------------------+--------------------+----------------------+----------
------------+
|            Model            |    Train Score     |     Test Score     |      R2 Score      | Mean Absolute Error  |  Mean Squ
ared Error  |
+-----------------------------+--------------------+--------------------+--------------------+----------------------+----------
------------+
|      Linear Regression      | 0.931462701459429  | 0.9385477788912998 | 0.9385477788912998 |  0.1433478813858459  | 0.0354261
5035818927  |
|       Lasso Regression      | 0.9311400666223492 | 0.9376395514555416 | 0.9376395514555416 |  0.144297792188092   |  0.035949
7278809885  |
|   Random Forest Regressor   | 0.9900485867151222 | 0.9844384476534296 | 0.9844384476534296 | 0.046774193548387105 | 0.0089709
67741935487 |
| Gradient Boosting Regressor | 0.9999898607888128 | 0.9896029904443768 | 0.9896029904443768 | 0.016883726664060645 | 0.0059936
97496165695 |
+-----------------------------+--------------------+--------------------+--------------------+----------------------+----------
------------+
```

Summery

- After visually analyzing the data we built the model using **Linear Regression, Lasso Regression, Random Forest Regression Gradient Boosting Regression.**
- From the table, we can see that the minimum mean square error is obtained by using **Gradient Boosting Regression** and the value is **0.017192401992716096**
- Gradient Boosting is good for predicting the lung cancer with 99.99% accuracy, quite amazing.

Summery (Cont..)

- It also shows us that obesity, air pollution, balanced diet, cough of blood, passive smoker is strongly correlated with lung cancer, therefore we should be alert when these factors occur to us.
- Best treatment for the cancer patient will be early check up if these symptoms are occurs; cough of blood, fatigue, dry cough, obesity, genetic risk and chest pain. And get treatment from doctor as soon as possible.
- We can prevent cancer by stop smoking, drinking alcohol, stay away from air polluted area, good balance diet and so on.