# Lung Cancer Detection using Machine Learning Algorithms

Gowthami Neelapu
ID - 23241a6644

## Abstract

Traditional lung cancer detection methods often rely on invasive procedures or costly tests, which can result in late diagnoses and higher mortality rates. This project aims to develop a machine learning model capable of identifying potential lung cancer cases using readily available, non-invasive categorical data, including patient demographics and medical history. We employ various algorithms, including Logistic Regression, Support Vector Classifier (SVC), Random Forest and Gradient Boosting to determine the most effective model for predicting the presence of lung cancer. The proposed detection model holds promise for significantly enhancing the early diagnosis of lung cancer by utilizing machine learning techniques.

## Introduction

Early detection plays a crucial role in improving survival rates, as timely diagnosis allows for more effective treatment options. However, traditional lung cancer detection methods, such as X-rays and biopsies, are often invasive, costly, and can delay diagnosis, leading to poorer patient outcomes.Our system uses machine learning algorithms to analyze patient data, detect patterns, and make predictions based on accessible features like smoking habits, medical history, symptoms like wheezing and chest pain, and other demographic factors. By incorporating features such as anxiety, chronic disease, fatigue, alcohol consumption, coughing, and swallowing difficulty, the system aims to identify individuals at a higher risk of developing lung cancer. The main algorithms used for prediction include the Random Forest and Gradient Boosting. The system is designed to work alongside existing diagnostic tools, providing additional insights and helping medical professionals make informed decisions. Ultimately, this technology could contribute to earlier detection and improved care for lung cancer patients, resulting in better overall outcomes.

## Problem Statement

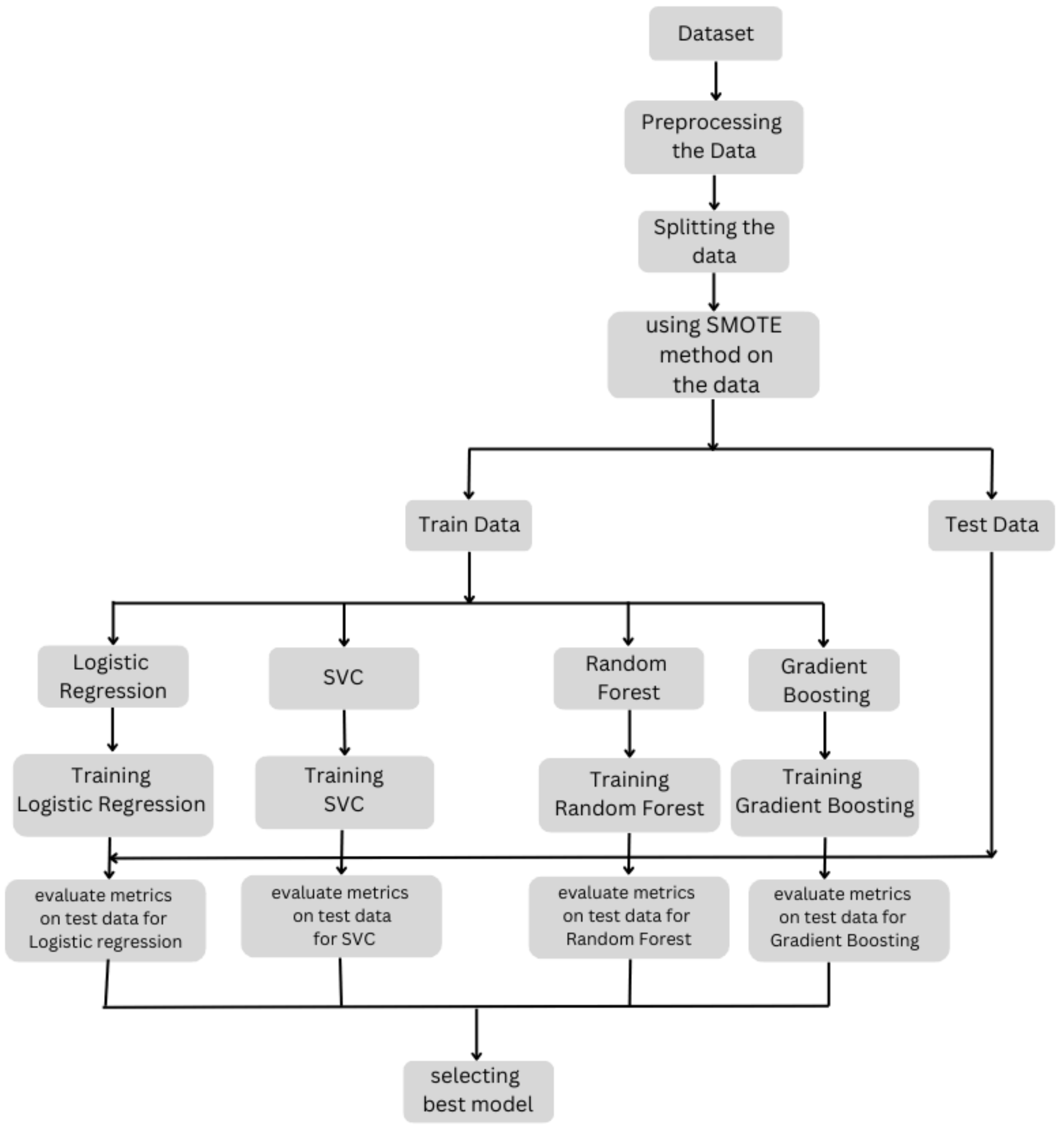
Early detection of Lung cancer plays a critical role in improving survival rates. However, identifying lung cancer based on patient data remains a challenge, especially when using easily accessible, categorical data such as smoking habits, family history, and other lifestyle factors. The project aims to predict the likelihood of lung cancer using machine learning models, given the complex relationships between these factors accurately by comparing multiple algorithms to find the most reliable prediction method.

## Objective

The primary objectives of this project are as follows:

1. Develop a Machine Learning Model: To create an effective machine learning model that can accurately predict the likelihood of lung cancer based on accessible categorical data.
2. Compare Multiple Algorithms: To evaluate and compare the performance of various machine learning algorithms, including Logistic Regression, Support Vector Machine (SVM), Random Forests and Gradient Boosting in predicting lung cancer cases.

## Flowchart


Dataset
Preprocessing the Data
Splitting the data
using SMOTE method on the data
Train Data
Test Data
Logistic Regression
SVC
Random Forest
Gradient Boosting
Training Logistic Regression
Training SVC
Training Random Forest
Training Gradient Boosting
evaluate metrics on test data for Logistic regression
evaluate metrics on test data for SVC
evaluate metrics on test data for Random Forest
evaluate metrics on test data for Gradient Boosting
selecting best model

## Dataset description

The dataset used for this lung cancer detection project was sourced from **Kaggle**, a well-known platform for data science and machine learning challenges.The dataset includes categorical data and 3000 user inputs of various pieces of information about patients, such as their background, medical history, lifestyle choices, and symptoms.

The key features of the dataset are:

- **GENDER**: The patient's gender.
- **AGE**: The patient's age.
- **SMOKING**: Whether or not the patient has a history of smoking.
- **YELLOW FINGERS**: If the patient has yellow fingers, often a sign of smoking.
- **ANXIETY**: Whether the patient suffers from anxiety.
- **PEER PRESSURE**: Whether the patient feels pressure from peers.
- **CHRONIC DISEASE**: If the patient has any chronic illnesses.
- **FATIGUE**: Whether the patient often feels tired.
- **ALLERGY**: If the patient has allergies.
- **WHEEZING**: Whether the patient has wheezing.
- **ALCOHOL CONSUMPTION**: Whether the patient drinks alcohol.
- **COUGHING**: Whether the patient has a persistent cough.
- **SHORTNESS OF BREATH**: If the patient has difficulty breathing.
- **SWALLOWING DIFFICULTY**: Whether the patient has trouble swallowing.
- **CHEST PAIN**: If the patient experiences chest pain.
- **LUNG CANCER**: The main outcome showing whether the patient has lung cancer (this is the target variable the model will predict).

## Algorithm used

After applying the **SMOTEENN** function from the imbalanced-learn library to handle imbalanced data, we proceeded with training our model. Below is a brief description of the algorithms we used:

1. **Logistic Regression**: This is a simple algorithm that helps in predicting whether an outcome is true or not (in this case, lung cancer or not). It works by analyzing the relationship between different features (like age, smoking, etc.) and the target variable (lung cancer). It's easy to understand and often performs well when the relationship between the data is straightforward.
2. **Support Vector Machines (SVM)**: SVM works by finding a boundary that best separates the data into two groups. It’s particularly useful for classification problems, like predicting lung cancer, because it can handle cases where the data isn’t perfectly separable. SVM also works well when the data has many features and isn’t easy to separate.
3. **Random Forest**: This algorithm creates multiple decision trees during training and then combines their predictions to get a more accurate result. It’s powerful because it reduces errors that a single decision tree might make, and it handles complex data

better by considering multiple features at once. It’s also good at avoiding overfitting (where the model becomes too specific to the training data).

4. **Gradient Boosting:** Gradient Boosting builds many small decision trees, each one fixing the mistakes of the previous ones. This step-by-step improvement makes it very powerful for complex problems, like detecting lung cancer. It works well even with noisy data and many features, but it needs careful tuning to avoid overfitting (learning too much from the training data).

## Motivation Behind Choosing These Algorithms

**Logistic Regression**:

- Simple to implement
- Provides a solid baseline for comparisons

**Random Forest**:

- Excels at finding complex patterns
- Reduces errors by using multiple decision trees

**Support Vector Machine (SVM)**:

- Effective at separating data that isn’t easily divided
- Suitable for lung cancer prediction due to its precision with challenging data

**Gradient Boosting**:

- Iteratively improves predictions by correcting previous model errors
- Powerful for handling complex data.

## Model Training and Evaluation

**Splitting the Dataset into Training and Testing Sets**:

- The dataset is split into two parts:
    - **Training set** (80%) to train the model
    - **Testing set** (20%) to evaluate performance

**Handling Imbalanced Data**:

- The **SMOTEENN** function from the imbalanced-learn library is used to handle imbalanced data.
- It works in two ways:
    - **SMOTE (Synthetic Minority Over-sampling Technique)**: Generates synthetic samples of the minority class (lung cancer cases) to balance the dataset.

   ○ **ENN (Edited Nearest Neighbors)**: Removes misclassified points, reducing noise from the majority class.

**Predicted Performance Metrics**:

- Accuracy
- Precision
- Recall (Sensitivity)
- F1 Score

## Code and Visualizations

## Code:

### IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from imblearn.combine import SMOTEENN
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification_report, confusion_matrix,accuracy_score, f1_score
```

### IMPORTING DATASET

```
df=pd.read_csv(r"C:\Users\neeso\OneDrive\Desktop\jpy\dataset_3000.csv")
```

### EDA

### (info)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   GENDER                 3000 non-null   object
 1   AGE                    3000 non-null   int64 
 2   SMOKING                3000 non-null   int64 
 3   YELLOW_FINGERS         3000 non-null   int64 
 4   ANXIETY                3000 non-null   int64 
 5   PEER_PRESSURE          3000 non-null   int64 
 6   CHRONIC_DISEASE        3000 non-null   int64 
 7   FATIGUE                3000 non-null   int64 
 8   ALLERGY                3000 non-null   int64 
 9   WHEEZING               3000 non-null   int64 
 10  ALCOHOL_CONSUMING      3000 non-null   int64 
 11  COUGHING               3000 non-null   int64 
 12  SHORTNESS_OF_BREATH    3000 non-null   int64 
 13  SWALLOWING_DIFFICULTY  3000 non-null   int64 
 14  CHEST_PAIN             3000 non-null   int64 
 15  LUNG_CANCER            3000 non-null   object
dtypes: int64(14), object(2)
memory usage: 375.1+ KB
```

**(null)**

```
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
```

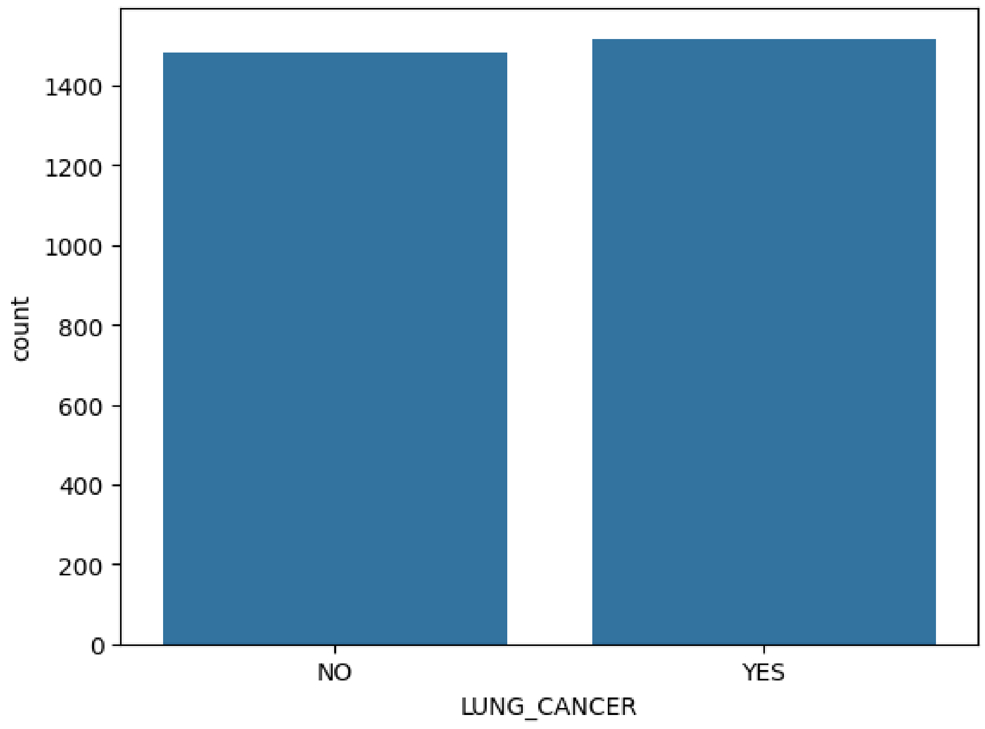
**(shape)**

```
df.shape
```

```
(3000, 16)
```

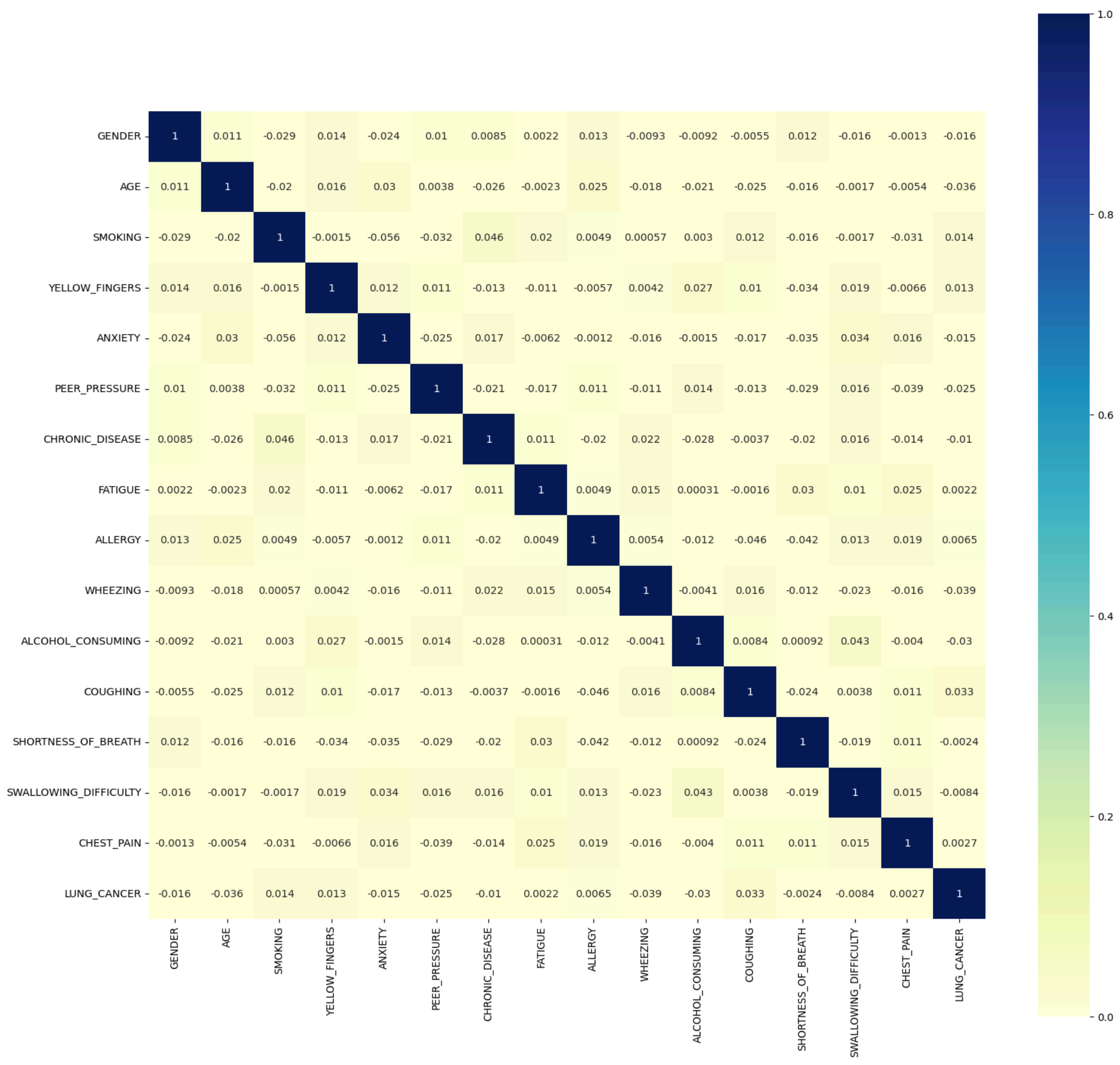
**(countplot)**

```
sns.countplot(x = 'LUNG_CANCER',data = df)
```



## (correlation)

```python
corrmat = df.corr()
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,annot=True, square=True, vmin=0, vmax=1,cmap="YlGnBu")
```



| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GENDER | 1 | 0.011 | -0.029 | 0.014 | -0.024 | 0.01 | 0.0085 | 0.0022 | 0.013 | -0.0093 | -0.0092 | -0.0055 | 0.012 | -0.016 | -0.0013 | -0.016 |
| AGE | 0.011 | 1 | -0.02 | 0.016 | 0.03 | 0.0038 | -0.026 | -0.0023 | 0.025 | -0.018 | -0.021 | -0.025 | -0.016 | -0.0017 | -0.0054 | -0.036 |
| SMOKING | -0.029 | -0.02 | 1 | -0.0015 | -0.056 | -0.032 | 0.046 | 0.02 | 0.0049 | 0.00057 | 0.003 | 0.012 | -0.016 | -0.0017 | -0.031 | 0.014 |
| YELLOW_FINGERS | 0.014 | 0.016 | -0.0015 | 1 | 0.012 | 0.011 | -0.013 | -0.011 | -0.0057 | 0.0042 | 0.027 | 0.01 | -0.034 | 0.019 | -0.0066 | 0.013 |
| ANXIETY | -0.024 | 0.03 | -0.056 | 0.012 | 1 | -0.025 | 0.017 | -0.0062 | -0.0012 | -0.016 | -0.0015 | -0.017 | -0.035 | 0.034 | 0.016 | -0.015 |
| PEER_PRESSURE | 0.01 | 0.0038 | -0.032 | 0.011 | -0.025 | 1 | -0.021 | -0.017 | 0.011 | -0.011 | 0.014 | -0.013 | -0.029 | 0.016 | -0.039 | -0.025 |
| CHRONIC_DISEASE | 0.0085 | -0.026 | 0.046 | -0.013 | 0.017 | -0.021 | 1 | 0.011 | -0.02 | 0.022 | -0.028 | -0.0037 | -0.02 | 0.016 | -0.014 | -0.01 |
| FATIGUE | 0.0022 | -0.0023 | 0.02 | -0.011 | -0.0062 | -0.017 | 0.011 | 1 | 0.0049 | 0.015 | 0.00031 | -0.0016 | 0.03 | 0.01 | 0.025 | 0.0022 |
| ALLERGY | 0.013 | 0.025 | 0.0049 | -0.0057 | -0.0012 | 0.011 | -0.02 | 0.0049 | 1 | 0.0054 | -0.012 | -0.046 | -0.042 | 0.013 | 0.019 | 0.0065 |
| WHEEZING | -0.0093 | -0.018 | 0.00057 | 0.0042 | -0.016 | -0.011 | 0.022 | 0.015 | 0.0054 | 1 | -0.0041 | 0.016 | -0.012 | -0.023 | -0.016 | -0.039 |
| ALCOHOL_CONSUMING | -0.0092 | -0.021 | 0.003 | 0.027 | -0.0015 | 0.014 | -0.028 | 0.00031 | -0.012 | -0.0041 | 1 | 0.0084 | 0.00092 | 0.043 | -0.004 | -0.03 |
| COUGHING | -0.0055 | -0.025 | 0.012 | 0.01 | -0.017 | -0.013 | -0.0037 | -0.0016 | -0.046 | 0.016 | 0.0084 | 1 | -0.024 | 0.0038 | 0.011 | 0.033 |
| SHORTNESS_OF_BREATH | 0.012 | -0.016 | -0.016 | -0.034 | -0.035 | -0.029 | -0.02 | 0.03 | -0.042 | -0.012 | 0.00092 | -0.024 | 1 | -0.019 | 0.011 | -0.0024 |
| SWALLOWING_DIFFICULTY | -0.016 | -0.0017 | -0.0017 | 0.019 | 0.034 | 0.016 | 0.016 | 0.01 | 0.013 | -0.023 | 0.043 | 0.0038 | -0.019 | 1 | 0.015 | -0.0084 |
| CHEST_PAIN | -0.0013 | -0.0054 | -0.031 | -0.0066 | 0.016 | -0.039 | -0.014 | 0.025 | 0.019 | -0.016 | -0.004 | 0.011 | 0.011 | 0.015 | 1 | 0.0027 |
| LUNG_CANCER | -0.016 | -0.036 | 0.014 | 0.013 | -0.015 | -0.025 | -0.01 | 0.0022 | 0.0065 | -0.039 | -0.03 | 0.033 | -0.0024 | -0.0084 | 0.0027 | 1 |

## LABEL ENCODING

```python
le = preprocessing.LabelEncoder()
col_to_encode = ['GENDER', 'LUNG_CANCER', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',
                 'CHRONIC_DISEASE', 'FATIGUE', 'ALLERGY', 'WHEEZING', 'ALCOHOL_CONSUMING',
                 'COUGHING', 'SHORTNESS_OF_BREATH', 'SWALLOWING_DIFFICULTY', 'CHEST_PAIN']
for cols in col_to_encode:
    df[cols] = le.fit_transform(df[cols])
```

## MODEL BUILDING

```
scaler = StandardScaler()
```

```
X = df.drop('LUNG_CANCER', axis = 1)
y = df['LUNG_CANCER']
```

```
smote = SMOTEENN()
X_resampled, y_resampled = smote.fit_resample(X,y)
```

```
xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)
```

## TRAINING ALGORITHMS

## (LOGISTIC REGRESSION)

```
lr_model=LogisticRegression()
```

```
lr_model.fit(xr_train,yr_train)
```

```
LogisticRegression
LogisticRegression()
```

```
yrl_predict = lr_model.predict(xr_test)

print("classification ")
print(classification_report(yr_test, yrl_predict))

print("accuracy score = ", accuracy_score(yr_test, yrl_predict))
```

```
classification
              precision    recall  f1-score   support

           0       0.61      0.68      0.64        37
           1       0.68      0.61      0.64        41

    accuracy                           0.64        78
   macro avg       0.64      0.64      0.64        78
weighted avg       0.64      0.64      0.64        78

accuracy score =  0.6410256410256411
```

## (SVC)

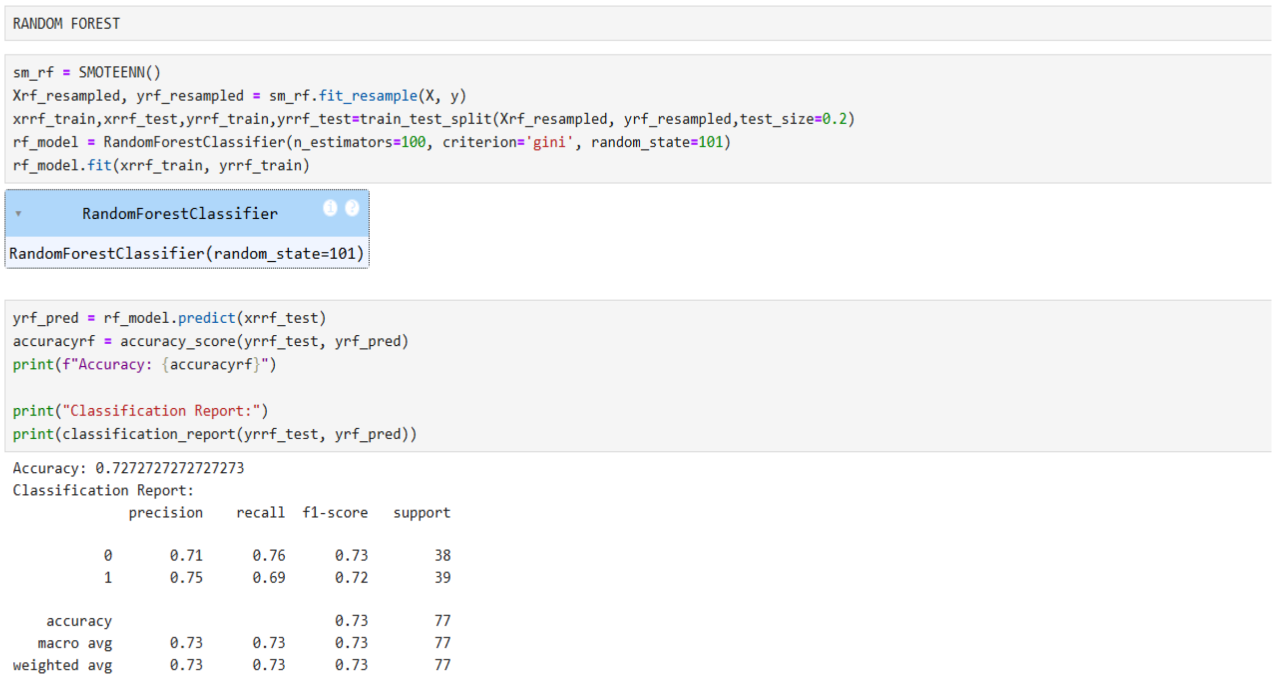
```
svc_model_smote = SVC(kernel='rbf', random_state=42)
svc_model_smote.fit(xr_train,yr_train)
```

```
SVC
SVC(random_state=42)
```

```
yrs_predict = svc_model_smote.predict(xr_test)

print("classification ")
print(classification_report(yr_test, yrs_predict))
print("accuracy score = ", accuracy_score(yr_test, yrs_predict))
```

```
classification
              precision    recall  f1-score   support

           0       0.59      0.78      0.67        37
           1       0.72      0.51      0.60        41

    accuracy                           0.64        78
   macro avg       0.66      0.65      0.64        78
weighted avg       0.66      0.64      0.64        78

accuracy score =  0.6410256410256411
```

## (RANDOM FOREST)



```
RANDOM FOREST
```

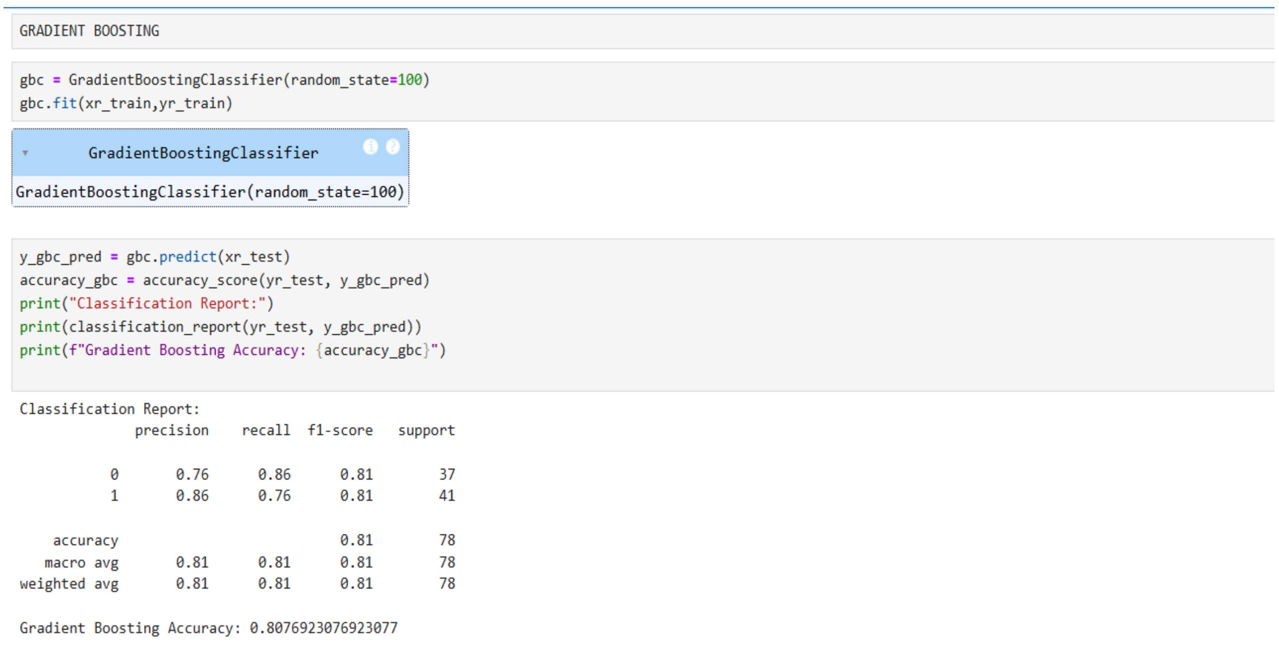
```
sm_rf = SMOTEENN()
Xrf_resampled, yrf_resampled = sm_rf.fit_resample(X, y)
xrrf_train,xrrf_test,yrrf_train,yrrf_test=train_test_split(Xrf_resampled, yrf_resampled,test_size=0.2)
rf_model = RandomForestClassifier(n_estimators=100, criterion='gini', random_state=101)
rf_model.fit(xrrf_train, yrrf_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=101)
```

```
yrf_pred = rf_model.predict(xrrf_test)
accuracyrf = accuracy_score(yrrf_test, yrf_pred)
print(f"Accuracy: {accuracyrf}")

print("Classification Report:")
print(classification_report(yrrf_test, yrf_pred))
```

```
Accuracy: 0.7272727272727273
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.76      0.73        38
           1       0.75      0.69      0.72        39

    accuracy                           0.73        77
   macro avg       0.73      0.73      0.73        77
weighted avg       0.73      0.73      0.73        77
```

## (GRADIENT BOOSTING)



```
GRADIENT BOOSTING
```

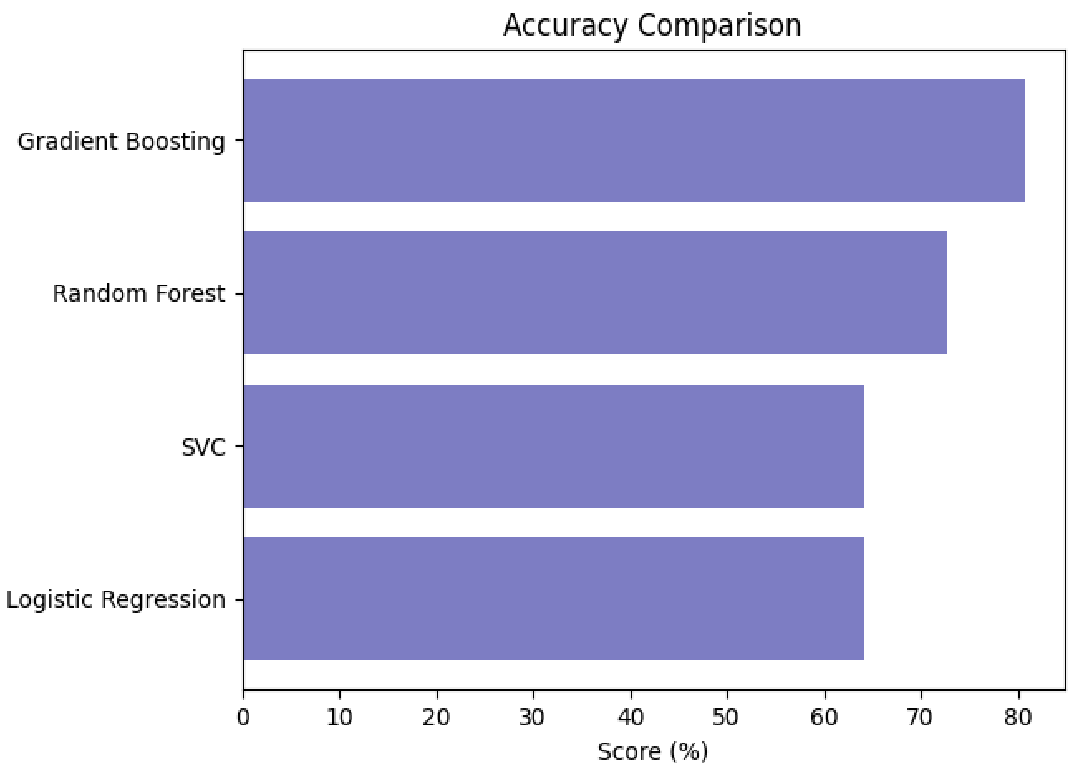
```
gbc = GradientBoostingClassifier(random_state=100)
gbc.fit(xr_train,yr_train)
```

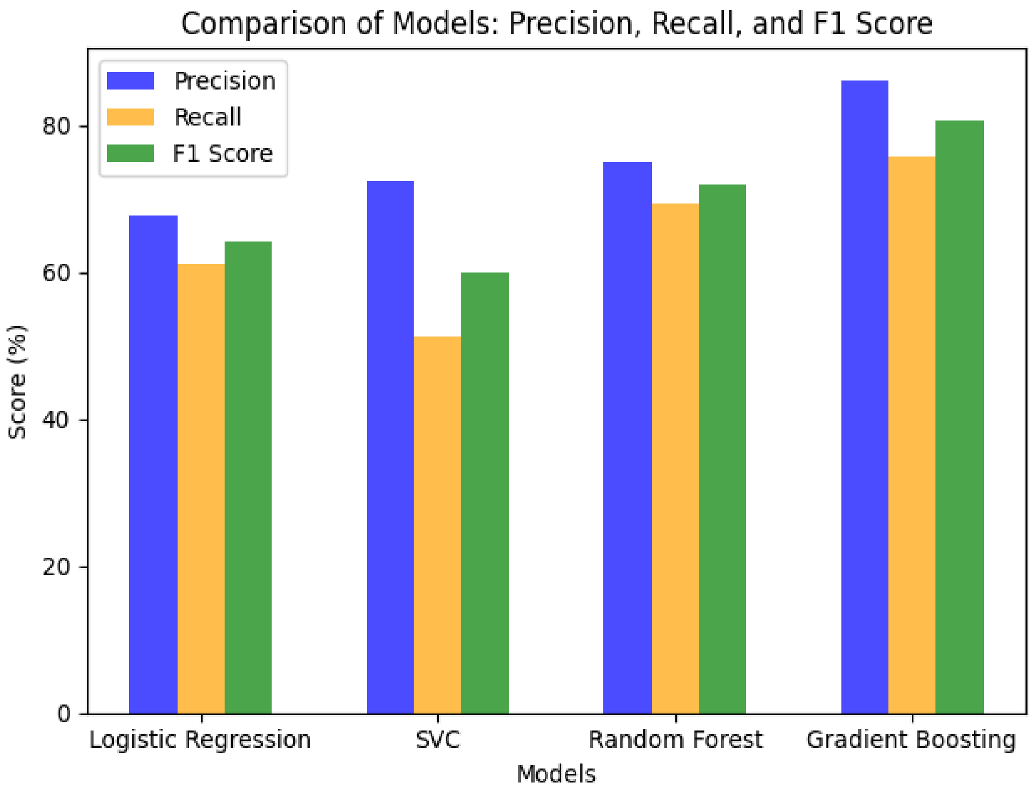
```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=100)
```

```
y_gbc_pred = gbc.predict(xr_test)
accuracy_gbc = accuracy_score(yr_test, y_gbc_pred)
print("Classification Report:")
print(classification_report(yr_test, y_gbc_pred))
print(f"Gradient Boosting Accuracy: {accuracy_gbc}")
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.86      0.81        37
           1       0.86      0.76      0.81        41

    accuracy                           0.81        78
   macro avg       0.81      0.81      0.81        78
weighted avg       0.81      0.81      0.81        78

Gradient Boosting Accuracy: 0.8076923076923077
```

**ACCURACY AND CLASSIFICATION REPORT COMPARISON**





## CONCLUSION

- Successfully developed a machine learning-based approach to predict lung cancer.
- Applied algorithms are **Logistic Regression**, **Support Vector Classifier (SVC)**, **Random Forest**, and **Gradient Boosting**.
- Used **SMOTEENN** to handle class imbalance in the dataset, resulting in promising model performance.

♦ **Model Comparison**:

- Based on **accuracy** and **classification report** comparison, the **best model** was **Gradient Boosting**, followed by **Random Forest**.

♦ **Balancing the Dataset**:

- To achieve better performance, **SMOTE** was used to balance the dataset.
- Dataset: The lung cancer detection dataset was obtained from Kaggle.

♦ Lung Cancer Dataset (kaggle.com)