**Lung Cancer Detection Using Machine Learning**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **SN** | **TITLES** | **Page No** |
| 1 | Introduction | 3 |
| 2 | Objective | 4 |
| 3 | Background | 4 - 5 |
| 4 | Hardware & Software Requirements | 6 |
| 5 | Coding | 6 - 13 |
| 6 | Screen Shots | 13 - 16 |
| 7 | Future Scope | 17 |
| 8 | Conclusion | 17 |
| 9 | Bibliography & References | 18 |

**Introduction**

Lung cancer is one of the main cause of the death and health issue in many countries with a 5-year survival rate of only 10-16%. People who smoke have the greatest risk of lung cancer, though lung cancer can also occur in people who have never smoked. The risk of lung cancer increases with the length of time and number of cigarettes you've smoked. If you quit smoking, even after smoking for many years, you can significantly reduce your chances of developing lung cancer. In this project we use machine learning algorithms to diagnose a cancer and start treatment in early stages. With the expected increase in the number of preventive/early-detection measures, scientists are working in computerized solutions that help alleviate the work of doctors, improve diagnostics' precision by reducing the subjectivity factor, speedup the analysis and reduce medical costs. In order to detect malignant nodules, specific features moved to be recognized and measured. Based on the detected features and their combination, cancer probability can be assessed. However, this task is very difficult, even for an experienced medical doctor, since nodule presence and positive cancer diagnosis are not easily related.

In this project I have use KNN & Decision Tree algorithms to predict the accuracy of the cancer.

In this project we use scikit-learn libraries like sk-learn and pandas to predict and classify the dataset of the lung cancer patients. Slicing the dataset and feature scaling options are used to train the dataset. After that we use confusion matrix, fl score and accuracy score to predict the accuracy of the result.

**Objectives-**

This project focuses on achieving superior accuracy in identifying lung cancer patients, addressing the urgent need for early detection of this highly fatal cancer. To meet this goal, we utilize the KNeighbors Classifier and Decision Tree algorithm for classification, optimizing their capabilities on the provided dataset.

Using machine learning techniques, our objective is to categorize individuals based on their lung cancer status, leveraging the 'Result' attribute in the dataset. A 'Result' value of 0 indicates the absence of lung cancer, while a value of 1 signifies its presence.

**Background-**

To perform KN algorithm and Decision Tree algorithm we use scikit-learn library. In the background of this project, used libraries are given below-

**Numpy** - NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

**Pandas** - Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

**Sklearn** - Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and Scily.

**KNeighbors Classifier** - KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. All training data usedin the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory.

**Decision Tree** - DTs are a non-parametric supervised learning method used

for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

**Dataset Information -**

The dataset contains 61 instances and 5 attributes which are-

Attributes Information-

* Age
* Smokes
* AreaQ
* Alcohol
* Result(0,1)

IDE used is Pycharm based on Python 3 interpreter.

**Hardware & Software Requirement**

|  |  |
| --- | --- |
| **Hardware Tools** | **Minimum Requirements** |
| **Processor** | **i5 or above** |
| **Ram** | **4GB** |
| **Monitor** | **15.6’’ colored** |
| **Mouse** | **optical** |
| **Keyboard** | **122 keys** |
| **Hard Disk** | **50Gb** |

|  |  |
| --- | --- |
| **Software Tools** | **Minimum Requirements** |
| **Platform** | **Windows** |
| **Operating System** | **Windows 8 or above** |
| **Technology** | **Machine Learning - Python** |
| **Scripting Language** | **Python** |
| **IDE** | **Pycharm** |

**Coding-**

import os

os.chdir("/Users/enaikeleomoh/Documents/mywork/python-1/ML/LUNGS CANCER PREDICTION”)

import numpy as np # Import the NumPy library for numerical computations

import pandas as pd # Import the Pandas library for data manipulation and analysis

from pandas. plotting import scatter\_matrix

import math

# Assuming you have imported the necessary libraries for KNN

from sklearn.neighbors import KNeighborsClassifier

import matplotlib.pyplot as plt # Import the Matplotlib library for data visualization

import seaborn as sns # Import the Seaborn library for statistical data visualization

from sklearn.preprocessing import StandardScaler

# Importing necessary libraries for evaluation metrics

from sklearn.metrics import roc\_curve, auc, recall\_score, precision\_score, accuracy\_score, f1\_score, confusion\_matrix

from sklearn.linear\_model import LogisticRegression # Import the LogisticRegression class for logistic regression

from sklearn.model\_selection import train\_test\_split # Import the train\_test\_split function for splitting data

from sklearn.metrics import confusion\_matrix # Import the confusion\_matrix function for evaluating classification results

from sklearn.metrics import classification\_report # Import the classification\_report function for detailed classification metrics

from sklearn.neighbors import KNeighborsClassifier # Import the KNeighborsClassifier class for k-nearest neighbors classification

from sklearn.tree import DecisionTreeClassifier # Import the DecisionTreeClassifier class for decision tree classification

from sklearn.ensemble import RandomForestClassifier # Import the RandomForestClassifier class for random forest classification

from sklearn.naive\_bayes import GaussianNB # Import the GaussianNB class for Gaussian naive Bayes classification

from sklearn.svm import SVC # Import the SVC class for support vector machine classification

# Importing our dataset

# Importing our dataset

print("Dataset:")

dataset = pd.read\_csv("lung\_cancer\_examples.csv")

print(len(dataset))

print(dataset.head())

scatter\_matrix(dataset)

plt.show()

# Selecting rows where 'Result' is 1

A = dataset[dataset['Result'] == 1]

# Selecting rows where 'Result' is 0

B = dataset[dataset['Result'] == 0]

# Creating scatter plots for A and B

plt.scatter(A['Age'], A['Smokes'], color="Black", label="1", alpha=0.4)

plt.scatter(B['Age'], B['Smokes'], color="Blue", label="0", alpha=0.4)

# Adding labels and title

plt.xlabel("Age")

plt.ylabel("Smokes")

plt.legend()

plt.title("Smokes vs Age")

# Display the plot

plt.show()

# Scatter plot for 'Alkhol' vs 'Age' for Result == 1

plt.scatter(A['Age'], A['Alkhol'], color="Black", label="1", alpha=0.4)

# Scatter plot for 'Alkhol' vs 'Age' for Result == 0

plt.scatter(B['Age'], B['Alkhol'], color="Blue", label="0", alpha=0.4)

# Adding labels and title

plt.xlabel("Age")

plt.ylabel("Alkhol")

plt.legend()

plt.title("Alkhol vs Age")

# Display the plot

plt.show()

# Scatter plot for 'Alkhol' vs 'Smokes' for Result == 1

plt.scatter(A['Smokes'], A['Alkhol'], color="Black", label="1", alpha=0.4)

# Scatter plot for 'Alkhol' vs 'Smokes' for Result == 0

plt.scatter(B['Smokes'], B['Alkhol'], color="Blue", label="0", alpha=0.4)

# Adding labels and title

plt.xlabel("Smokes")

plt.ylabel("Alkhol")

plt.legend()

plt.title("Smokes vs Alkhol")

# Display the plot

plt.show()

# Selecting features (x) and target variable (y) from the dataset

# We're extracting columns 3 to 4 for features (x) and column 6 for the target variable (y)

x = dataset.iloc[:, 3:5] # Features (independent variables)

y = dataset.iloc[:, 6] # Target variable (dependent variable)

# Splitting the dataset into training and testing sets

# We're using the train\_test\_split function from scikit-learn to divide the data

# random\_state is set to ensure reproducibility, and test\_size determines the proportion of data used for testing (20% in this case)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0, test\_size=0.2)

# Creating a StandardScaler object

sc\_x = StandardScaler()

# Scaling (standardizing) the features in the training set

x\_train = sc\_x.fit\_transform(x\_train)

# Scaling the features in the test set using the same scaling parameters as the training set

x\_test = sc\_x.transform(x\_test)

# starting to use the KNN algorithm

print("--------------\*\*\*\*Using KNN Algorithm\*\*\*\*-----------------")

# Assuming you have imported the necessary library for the KNN algorithm

from sklearn.neighbors import KNeighborsClassifier

# Importing the math module for mathematical operations

import math

# Calculating the square root of the number of training samples

a = math.sqrt(len(y\_train))

# Printing the result, which represents a commonly used parameter in the KNN algorithm

print(a)

# Define a KNN classifier with specific settings:

# - n\_neighbors: Number of neighbors to consider (here, 5 neighbors)

# - p: Parameter for the Minkowski distance (here, p=2 corresponds to Euclidean distance)

# - metric: Distance metric to use for the tree (here, 'euclidean' distance)

classifier = KNeighborsClassifier(n\_neighbors=5, p=2, metric='euclidean')

# Train (fit) the KNN model using the training data

classifier.fit(x\_train, y\_train)

# Use the trained model to make predictions on the test set

y\_pred = classifier.predict(x\_test)

# Print the predicted labels for the test set

print(y\_pred)

# Evaluating model

# ROC curve

# This section calculates and plots the Receiver Operating Characteristic (ROC) curve. ROC curves help visualize the performance of a classification model.

# fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

# roc\_auc = auc(fpr, tpr)

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.format(roc\_auc))

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc="lower right")

plt.show()

# Recall

# Recall, or sensitivity, is the ability of a classification model

# to identify all relevant instances in the dataset.

# Recall is the proportion of actual positive cases that were correctly identified by the model.

recall = recall\_score(y\_test, y\_pred)\*100

print('Recall: {:.2f}'.format(recall))

# Precision

# This section calculates and prints precision,

# which is the proportion of predicted positive cases

# that were correctly identified by the model.

# Precision is the ability of a classification model to return only relevant instances.

precision = precision\_score(y\_test, y\_pred)\*100

print('Precision: {:.2f}'.format(precision))

# Accuracy

# The accuracy of the model is calculated and printed.

# Accuracy is the proportion of correctly predicted instances out of the total instances.

# it the ratio of correctly predicted observation to the total observations.

accuracy = accuracy\_score(y\_test, y\_pred)\*100

print('Accuracy: {:.2f}'.format(accuracy))

# F1 Score

# F1 Score is the harmonic mean of precision and recall, providing a balance between the two.

f1 = f1\_score(y\_test, y\_pred) \*100

print('F1 Score: {:.2f}'.format(f1))

# Confusion Matrix(evaluate the performance of a classification model)

# The confusion matrix provides a summary of the model's

# performance, showing the number of true positives,

# true negatives, false positives, and false negatives.

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(cm)  
  
  
  
  
  
# Using Decision Tree Algorithm

# Create a Decision Tree classifier

classifier\_dt = DecisionTreeClassifier()

# Fit (train) the Decision Tree model on the training data

classifier\_dt.fit(x\_train, y\_train)

# Print a message indicating the start of the Decision Tree algorithm

print("-----------------\*\*\*\*Using Decision Tree Algorithm\*\*\*\*----------------")

# Calculate the accuracy on the training set

# This measures how well the model performs on the data it was trained on

accu\_train = np.sum(classifier\_dt.predict(x\_train) == y\_train) / float(len(y\_train))

print('Classification accuracy on train:', accu\_train \* 100)

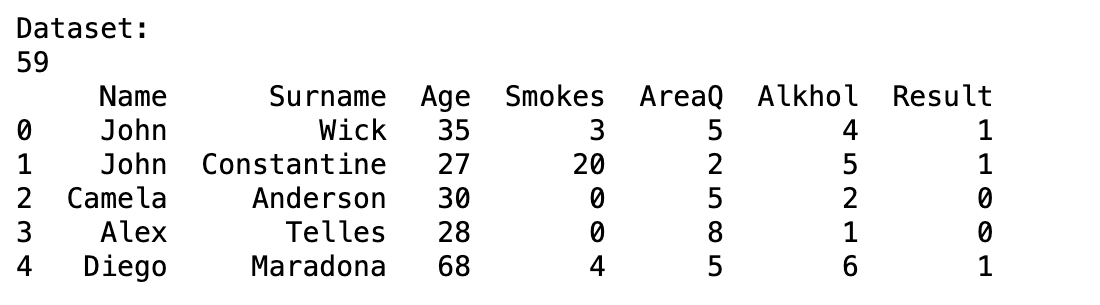
# Calculate the accuracy on the test set

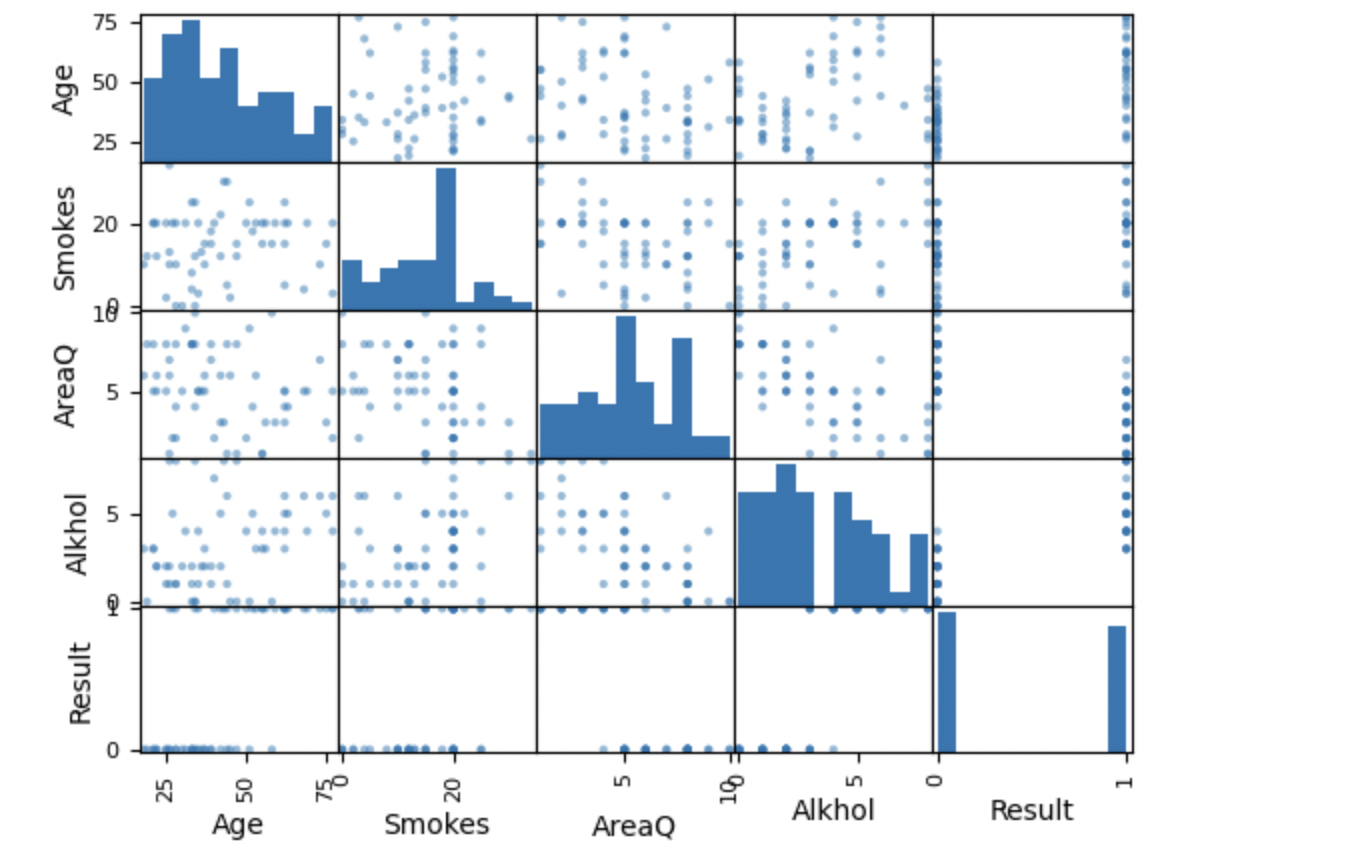
# This measures how well the model generalizes to new, unseen data

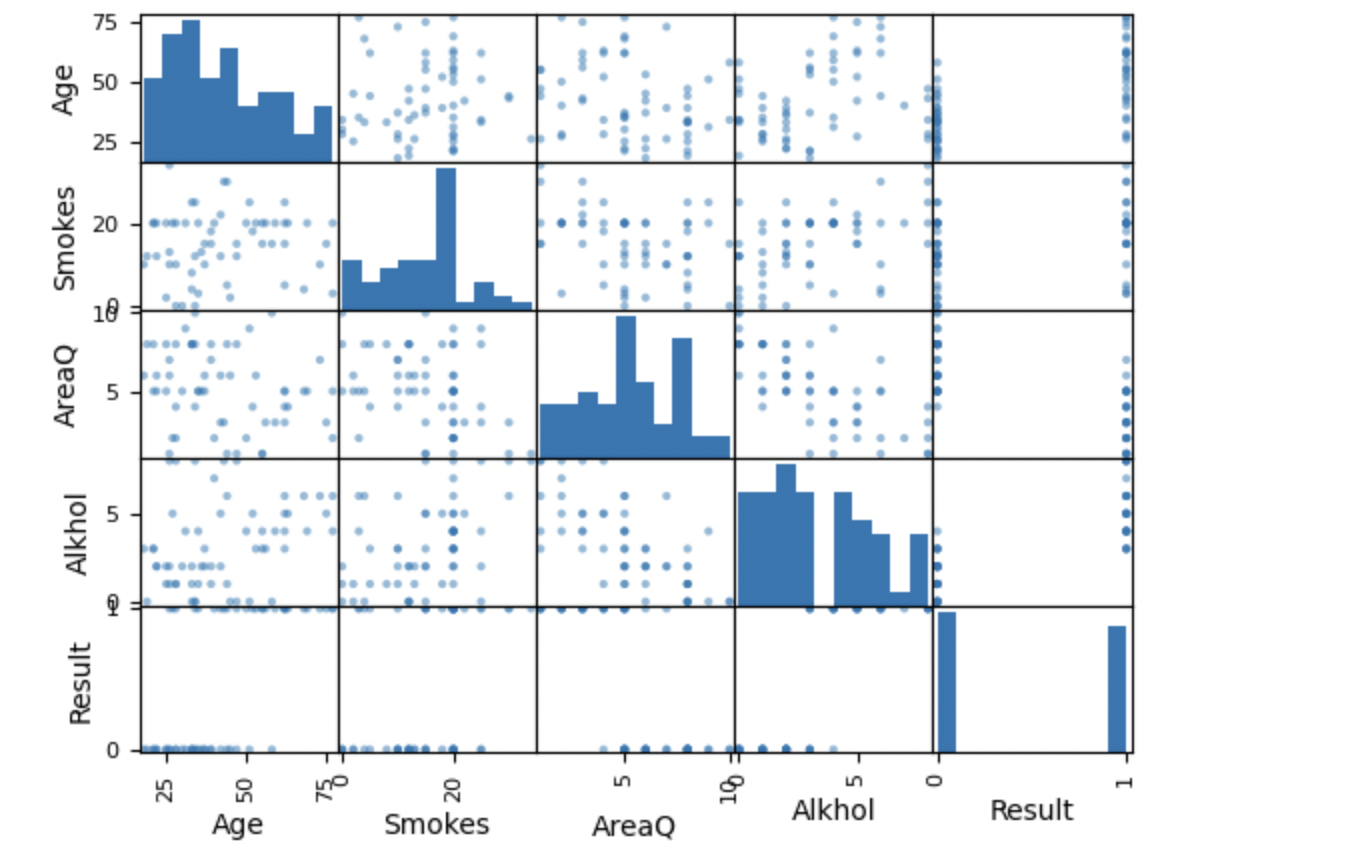
accu\_test = np.sum(classifier\_dt.predict(x\_test) == y\_test) / float(len(y\_test))

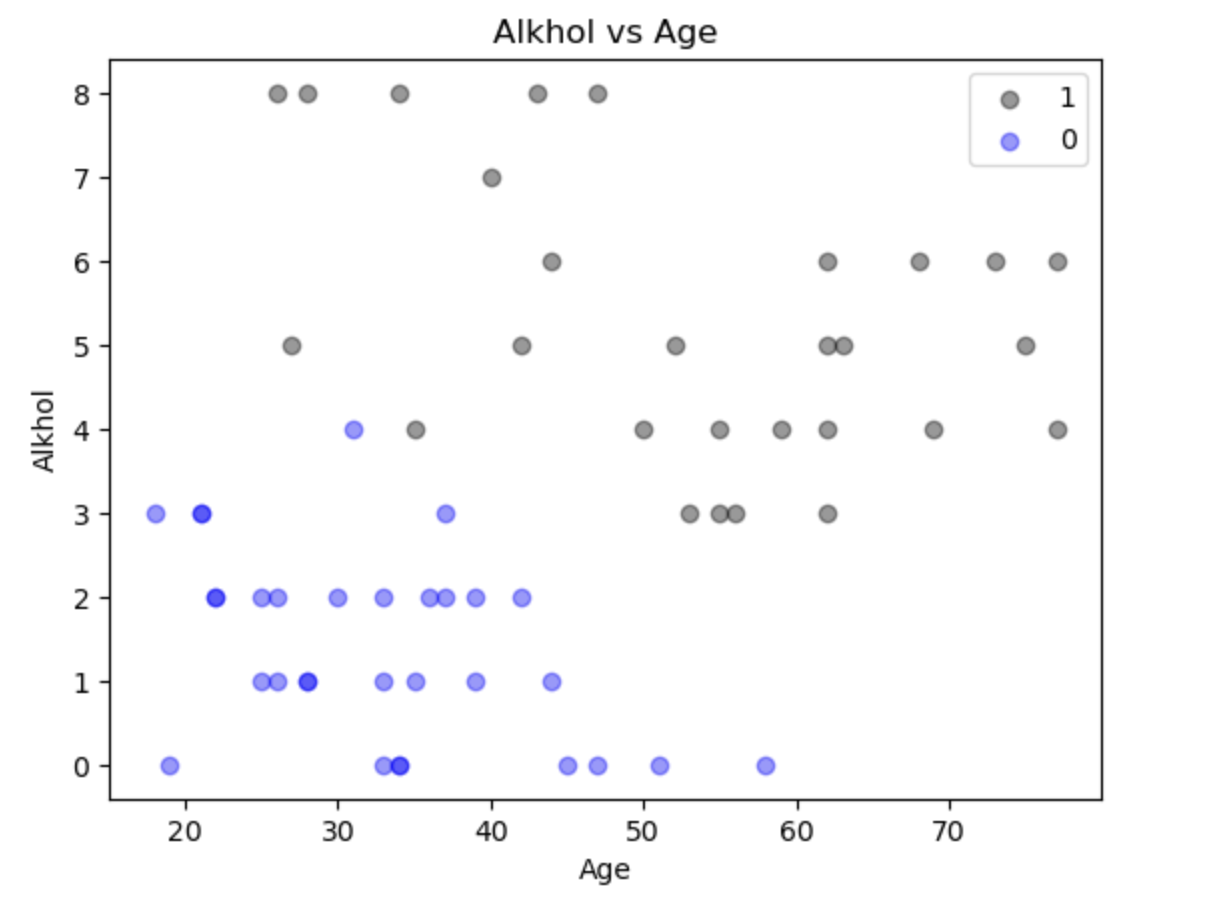
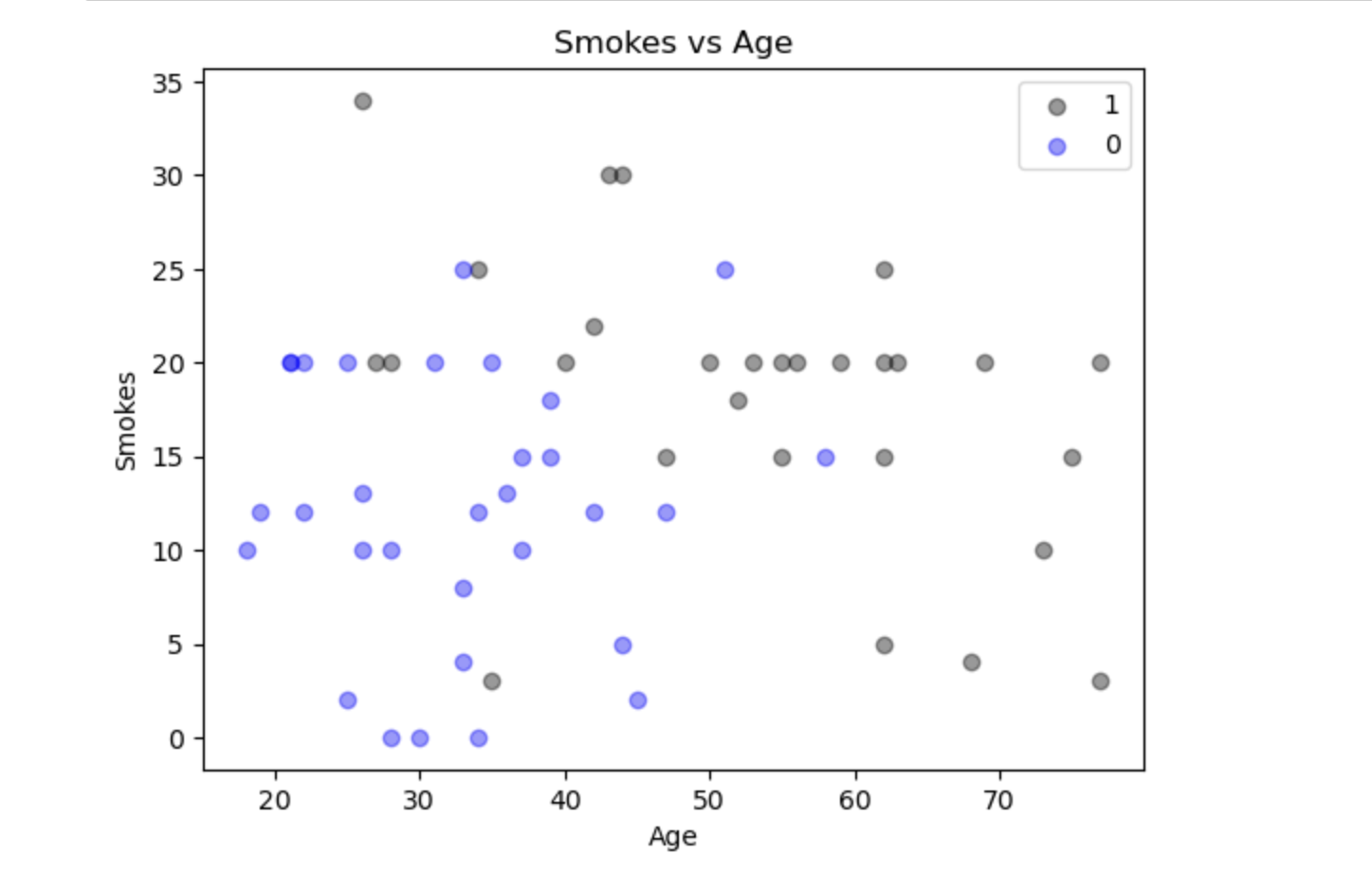
print('Classification accuracy on test:', accu\_test \* 100)

**SCREENSHOTS**

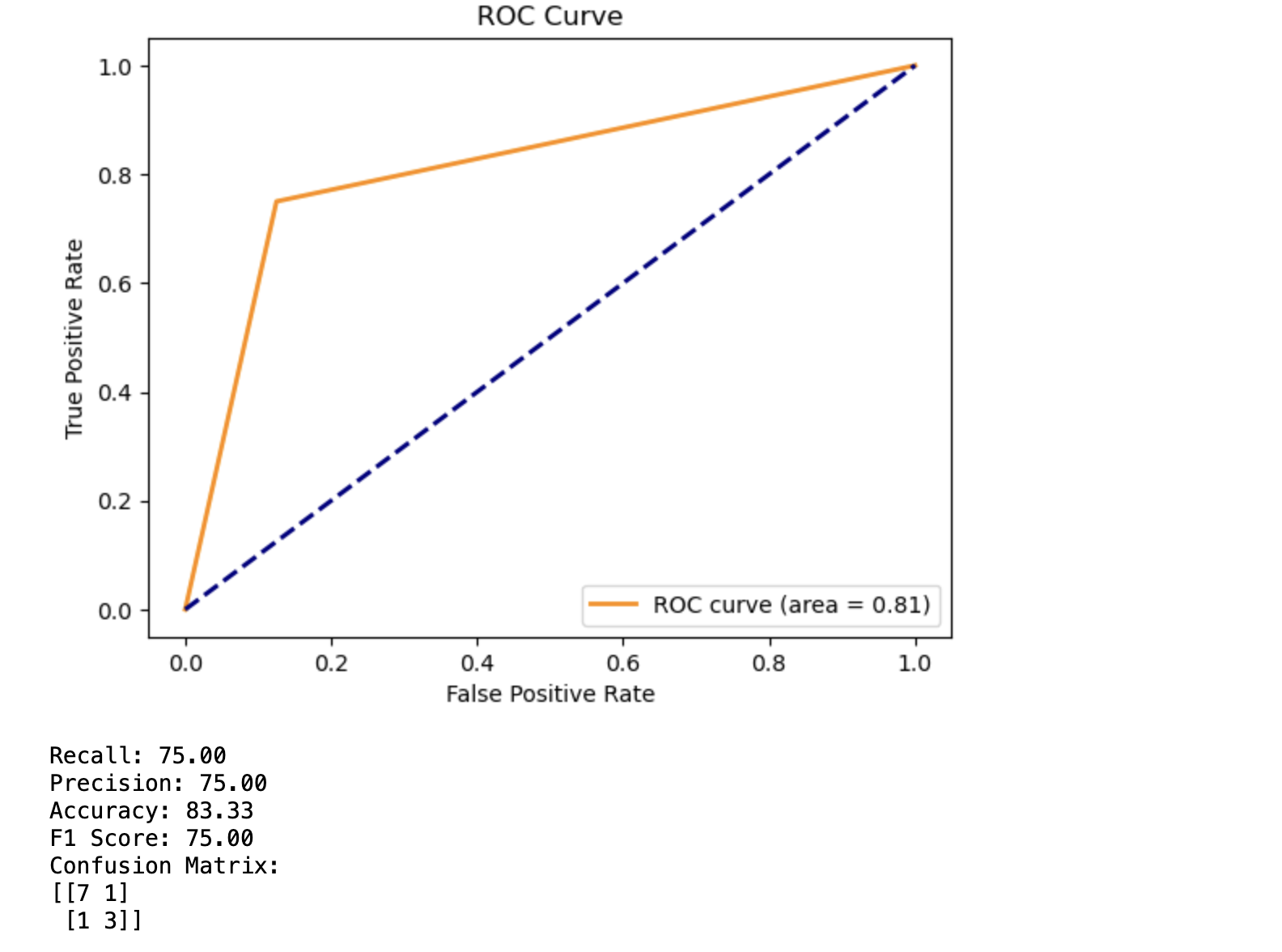
****



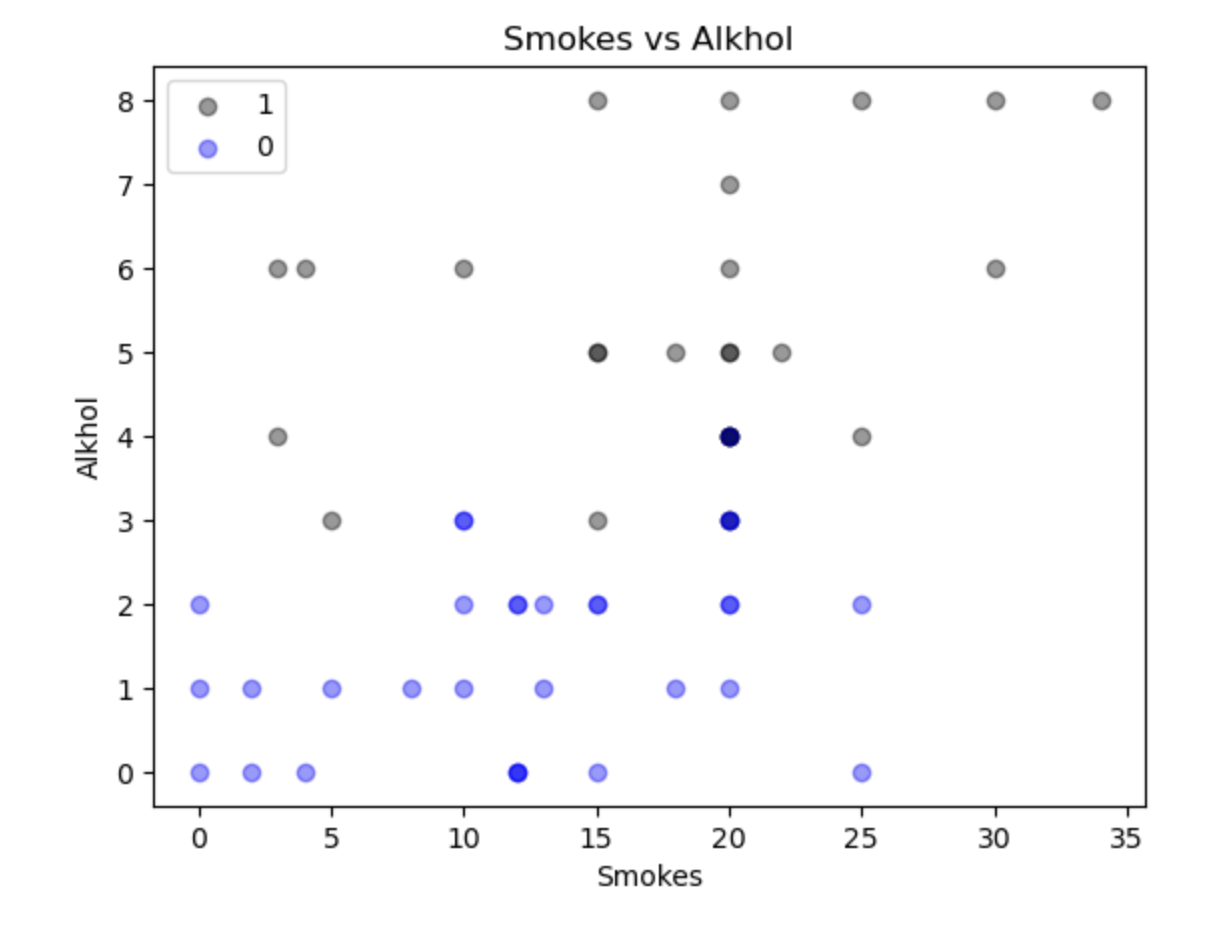




**OUTPUT**

****

**Future Scope:**

The lung cancer detection system utilizing machine learning techniques is highly efficient and provides improved results for radiologists, aiding in accurate patient detection. This system can be further enhanced with additional features to support radiologists and improve overall performance.

Machine learning plays a crucial role in enabling Artificial Intelligence, marking the future of healthcare. The healthcare sector is becoming increasingly data-driven, leveraging big data and machine learning to bring about significant advancements. These technologies not only enhance treatment and diagnosis options but also empower individuals to take control of their health through access to smarter healthcare.

**Conclusion:**

In this study, we processed the dataset to differentiate affected patients and determine the level of cancer growth using a machine learning system. The approach presented here aims to achieve the best accuracy in cancer diagnosis, assisting radiologists and paving the way for future enhancements. Future work should focus on improving classification accuracy through experiments with various alternatives.

Benchmarking the most performing architectures on available datasets using similar metrics can facilitate comparative analysis. One current limitation is the imbalanced nature of the data. The incorporation of new loss functions, designed to address the issue of unbalanced classes, such as focal loss, could enhance existing results and contribute to more efficient training.

With an increased dataset and improved data balance, better results can be achieved. This highlights the potential for continued advancements in lung cancer detection and the importance of ongoing research to refine and optimize machine learning models for enhanced healthcare outcomes.

**Bibliography & References-**

Dataset - Lung Cancer Dataset (<https://www.kaggle.com/datasets/yusufdede/lung-cancer-dataset?resource=download>)

Machine Learning Journal

ML Algorithms (machinelearningmastery.com)

Scikit-Learn Library (scikit-learn.org)

Lung Cancer Detection (www.kaggle.com)