Assignment

**Lung Cancer Prediction and Data Visualization**

**Aim:**

Aimed to develop and optimize machine learning models for predicting lung cancer risk. We explored the performance of logistic regression and AdaBoost classifier models, leveraging techniques like hyperparameter tuning with GridSearchCV to enhance their predictive accuracy. By evaluating the models on test datasets and comparing their performance metrics, including accuracy, precision, recall, and F1-score, we identified the best-performing model. Our findings suggest that with careful model selection and hyperparameter optimization, it's possible to develop robust predictive models with practical applications in healthcare settings.

Furthermore, we discussed the interpretability of the models and their potential for providing valuable insights into factors influencing lung cancer risk. This project serves as a foundation for future research aimed at refining these models and exploring additional avenues for improving predictive accuracy and generalization to diverse datasets.

**Data Info:** This dataset contains information on patients with lung cancer, including their age, gender, air pollution exposure, alcohol use, dust allergy, occupational hazards, genetic risk, chronic lung disease, balanced diet, obesity, smoking status, passive smoker status, chest pain, coughing of blood, fatigue levels, weight loss, shortness of breath, wheezing, swallowing difficulty, clubbing of finger nails, frequent colds, dry coughs, and snoring.

•Age: The age of the patient. (Numeric)

•Gender: The gender of the patient. (Categorical)

•Air Pollution: The level of air pollution exposure of the patient. (Categorical)

•Alcohol use: The level of alcohol use of the patient. (Categorical)

•Dust Allergy: The level of dust allergy of the patient. (Categorical)

•OccuPational Hazards: The level of occupational hazards of the patient. (Categorical)

•Genetic Risk: The level of genetic risk of the patient. (Categorical)

•chronic Lung Disease: The level of chronic lung disease of the patient. (Categorical)

•Balanced Diet: The level of balanced diet of the patient. (Categorical)

•Obesity: The level of obesity of the patient. (Categorical)

•Smoking: The level of smoking of the patient. (Categorical)

•Passive Smoker: The level of passive smoker of the patient. (Categorical)

•Chest Pain: The level of chest pain of the patient. (Categorical)

•Coughing of Blood: The level of coughing of blood of the patient. (Categorical)

•Fatigue: The level of fatigue of the patient. (Categorical)

•Weight Loss: The level of weight loss of the patient. (Categorical)

•Shortness of Breath: The level of shortness of breath of the patient. (Categorical)

•Wheezing: The level of wheezing of the patient. (Categorical)

•Swallowing Difficulty: The level of swallowing difficulty of the patient. (Categorical)

•Clubbing of Finger Nails: The level of clubbing of finger nails of the patient. (Categorical)

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**First**: Importing several helpful packages, Initial Analysis and load the dataset

**A screenshot of a computer

Description automatically generatedSplitting Into y and X:**

**Second:** **Dataset Visualization**

1. **plot creates a pie chart representing the distribution of levels (or classes) in the training dataset**

**A screenshot of a computer

Description automatically generated**

1. A blue and white squares

   Description automatically generated**Correlation plot** is a quick and effective tool for identifying linear relationships between quantitative variables. It utilizes correlation coefficients, such as Pearson's, to quantify the strength and direction of these relationships.

**Code:**

plt.figure(figsize=(20,15))

sns.heatmap(df.corr(), annot=True, cmap=plt.cm.PuBu)

plt.show()

1. **A group of graphs on a white background

   Description automatically generatedA group of graphs with different colored lines

   Description automatically generatedgrid of histograms with fitted Gaussian & grid of scatter plots with regression line**

**A diagram of a diagram

Description automatically generated with medium confidenceA graph with blue lines

Description automatically generatedThird:** Exploratory Data Analysis (EDA) to gain insights into the dataset's characteristics, distributions, and relationships

-Univariate Analysis (Distribution of Age) -Bivariate Analysis (Relationship between Age and Level)

A screenshot of a computer

Description automatically generated**Fourth:** **Model Building**

- **The training-testing** **data splitting**, which is a crucial step in machine learning model development to assess the model's performance on unseen data.

**First model:** **AdaBoost Classifier Model**

AdaBoost classifier along with cross-validation scores for logistic regression.

1. AdaBoost Classifier Model:

- An AdaBoost classifier (abc) is instantiated with parameters such as the number of estimators (n\_estimators), learning rate (learning\_rate), algorithm (algorithm), and random state (random\_state).

- The base estimator for AdaBoost is set to a decision tree classifier (DecisionTreeClassifier).

- The classifier is trained on the training data (X\_train and y\_train) using the (fit) method.

2. Model Evaluation:

- Predictions are made on the testing set (X\_test) using the trained AdaBoost classifier's predict method.

- The classification report is printed using classification\_report from sklearn.metrics. It provides metrics such as precision, recall, and F1-score for each class.

- A confusion matrix is plotted using confusion\_matrix from sklearn.metrics. It shows the counts of true positive, true negative, false positive, and false negative predictions.

3. Cross-validated scores for Logistic Regression:

- Cross-validated scores for logistic regression are computed using cross\_val\_score from sklearn.model\_selection with 5-fold cross-validation.

- The cross-validated scores are printed, and the mean cross-validated score is calculated.

A blue squares with white text

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Second model:** **Logistic Regression Model**

1. Data Preparation:

- The Level column in the DataFrame df is replaced with numerical values (1 for Low, 2 for Medium, and 3 for High). This is likely done to facilitate the classification task since many machine learning algorithms require numerical inputs.

- The Patient Id column is dropped from the features (X) as it is not used for prediction.

- Features (X) and labels (y) are separated.

2. Data Splitting:

- The data is split into training and testing sets using the train\_test\_split function from sklearn.model\_selection. The testing set size is set to 30% of the data, and a random state is specified for reproducibility.

3. Logistic Regression Model Training:

- A logistic regression model is created using LogisticRegression from sklearn.linear\_model.

- The model is trained on the training data (X\_train and y\_train) using the fit method.

4. Model Evaluation:

- Predictions are made on the testing set (X\_test) using the trained model's predict method.

- The classification report is printed using classification\_report from sklearn.metrics. It provides metrics such as precision, recall, and F1-score for each class.

- A confusion matrix is plotted using confusion\_matrix from sklearn.metrics. It shows the counts of true positive, true negative, false positive, and false negative predictions.

A blue squares with white text

Description automatically generated - Cross-validated scores are computed using cross\_val\_score from sklearn.model\_selection with 5-fold cross-validation. This provides an estimate of the model's performance on unseen data.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Hyperparameter Tuning**

1. Define Hyperparameter Grid: The param\_grid dictionary defines the grid of hyperparameters to search over. It includes regularization parameter C, solver for optimization problem, and maximum number of iterations for the logistic regression model.

2. Create GridSearchCV Object: GridSearchCV is a function in scikit-learn that performs exhaustive search over a specified parameter grid to find the best parameters for a model. In this code, it's used to search for the best combination of hyperparameters for the logistic regression model.

3. Fit Grid Search to Data: The fit method of the GridSearchCV object is called to fit the grid search to the training data (X\_train and y\_train). This will train and evaluate the logistic regression model for each combination of hyperparameters specified in the param\_grid.

4. Print Best Hyperparameters: After fitting the grid search, the best hyperparameters found during the search are printed.

5. Get Best Model: The best model found during the grid search, based on the specified scoring metric (accuracy in this case), is extracted using the best\_estimator attribute of the GridSearchCV object.

6. Evaluate Best Model: The best model is evaluated on the test data (X\_test) using the predict method to generate predictions. These predictions are then used to calculate a classification report and a confusion matrix, which are printed and visualized, respectively.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Based on the classification report, the best logistic regression model achieved high precision, recall, and F1-score for each class, as well as a high overall accuracy of 99%. Therefore, the model appears to perform very well on the test dataset, with strong predictive performance across all classes.

**Conclusion**

In conclusion, our study aimed to develop and optimize machine learning models for predicting lung cancer risk using a dataset containing various patient attributes. We explored the performance of two models, logistic regression and AdaBoost classifier, leveraging techniques like hyperparameter tuning with GridSearchCV to enhance predictive accuracy.

Our findings revealed that both models achieved high accuracy, precision, recall, and F1-scores, indicating their effectiveness in predicting lung cancer risk. The logistic regression model, after hyperparameter tuning, demonstrated exceptional performance, achieving an accuracy of 99% on the test dataset. These results highlight the potential of machine learning algorithms in healthcare settings, offering valuable insights into factors influencing lung cancer risk and paving the way for future research to refine and improve predictive models further. Overall, our study underscores the importance of leveraging advanced analytics to augment medical decision-making and improve patient outcomes in preventive healthcare.

**Deployment (**[**http://localhost:8501/**](http://localhost:8501/)**)**

import joblib

file='lungcancer'

joblib.dump(classifier,"lungcancer")

model=joblib.load(open("lungcancer",'rb'))

import joblib libery

the classifier object, which represents the trained logistic regression model, is saved to a file named lungcancer using the joblib.dump() function. This file will contain all the necessary information about the trained model.

1. Imports: These lines import the necessary libraries and modules needed for the application.

import streamlit as st

import requests

import pandas as pd

from streamlit\_option\_menu import option\_menu

import streamlit\_lottie as st\_lottie

import joblib

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import PIL as Image

2. Streamlit Page: This sets the title, icon, and initial state of the sidebar for the Streamlit app.

st.set\_page\_config(

    page\_title='lung cancer',

    page\_icon=':gem:',

    initial\_sidebar\_state='collapsed'

)

This function load\_lottie(url) is designed to fetch a Lottie animation file from a given URL and return its JSON content.

def load\_lottie(url):

    r = requests.get(url)

    if r.status\_code != 200:

        return None

    return r.json()

3. Load Model: This loads the pre-trained machine learning model from the file named "lungcancer".

model=joblib.load(open("lungcancer",'rb'))

4. Define Prediction Function: This function takes input features (age, gender, smoking, alcohol\_use) and uses the loaded model to make predictions.

def predict(age, gender, smoking, alcohol\_use):

    features = np.array([age, gender, smoking, alcohol\_use] + [0]\*20).reshape(1, -1)

    prediction = model.predict(features)

    return prediction

5. Sidebar Menu: This creates a sidebar menu with options for the user to navigate through the app.

ith st.sidebar:

    choose = option\_menu( None, ["Home", "Graphs", "About", "Contact"],

        icons=["house", "kanban", "book", "person\_lines"],

        menu\_icon="app-indicator",default\_index=0,

        styles={

            "container": {"padding": "5px !important", "background-color": "#fafafa"},

            "icon": {"color": "#EEEFFF", "font-size": "25px"},

            "nav-link": {"font-size": "16px", "text-align": "left"},

            "nav-link-selected": {"background-color": "#02ab21"}

        }

    )

6. Home Page: This section of the code displays the main page of the app where users can input their details and make predictions.

if choose == "Home":

    st.write('# Lung cancer prediction')

    st.subheader('Enter your details to lung cancer prediction')

    # User input

    age= st.number\_input("Enter your age:", min\_value=0)

    gender= st.radio("Select your gender:", ('male', 'female'))

    gender\_encoder = 1 if gender == 'male' else 0

    smoking= st.number\_input("Enter your smoking times per day:", min\_value=0)

    alcohol\_use= st.number\_input("Enter your alcohol use per day:", min\_value=0)

    #predict

    sample=predict(age,gender\_encoder,smoking,alcohol\_use)

    if st.button("predict"):

        if sample <= 4:

            st.write('This indicates a low level of lung cancer.')

    else:

        st.write('This indicates a high level of lung cancer.')

7. \*\*Graphs, About, Contact Pages\*\*: These sections define the content of the respective pages.

elif choose == "Graphs":

    st.write("# Graphs")

    st.image("download.png")

elif choose == "About":

    st.write("# About")

    st.write("This app provides a lung cancer prediction based on user input.")

elif choose == "Contact":

    st.write("# Contact")

    st.write("For inquiries, please contact us at a.yasser2160@nu.edu.eg")

A screenshot of a computer

Description automatically generated

**The App**