PROJECT REPORT

on

Lung Cancer Data Prediction

Submitted in partial fulfillment of requirements for the award of

IBM Internship (Data Science)

Submitted by:

Rishik Pathak (CS22BCAGN003)

Priyanku Gogoi (CS22BCAGN028)

Amit Karmakar (CS22BCAGN006)

Pranjeet Gogoi (CS22BCAGN013)

Minmoy Khound (CS22BCAGN007)

Department of Information Technology
School of Computing Sciences
The Assam Kaziranga University, Jorhat, Assam

MAR 2025

SCHOOL OF COMPUTING SCIENCES
THE ASSAM KAZIRANGA UNIVERSITY
JORHAT-785006 :: ASSAM :: INDIA

Abstract

Lung cancer is one of the most dangerous diseases in the world, and detecting it early can save lives. This project analyzes a lung cancer dataset to understand its key factors and develop models that help predict diagnoses. We used Python's data science tools, including pandas, matplotlib, and scikit-learn, to clean the data, explore patterns, and visualize important trends. Machine learning models were then trained to classify whether a person has lung cancer based on specific features. The project aims to improve awareness of lung cancer risks and demonstrate how data science can assist in early detection and diagnosis.

Introduction

Lung cancer is one of the deadliest diseases worldwide, and early detection plays a crucial role in improving patient outcomes. This project focuses on analyzing a lung cancer dataset to gain insights into its prevalence, contributing factors, and potential prediction models. Using Python's data science libraries like pandas, matplotlib, and scikit-learn, we performed exploratory data analysis (EDA), cleaned the dataset, visualized key trends, and built predictive models to classify whether a person is diagnosed with lung cancer.

Objective of the project

The main goal of this project is to study lung cancer data and build models to predict diagnoses. First, we analyze the data to find patterns, trends, and connections between different factors that may contribute to lung cancer. This helps us understand which features are important. Then, we use machine learning techniques to train models that can predict whether a person has lung cancer based on these features. By doing this, we aim to improve early detection and raise awareness about the factors linked to lung cancer.

Dataset Overview

The dataset used for this project is stored in lung_cancer_prediction_dataset.csv. It contains several features such as age, smoking habits, air pollution exposure, occupational exposure, and more. The target variable is Lung_Cancer_Diagnosis, which indicates whether a person has been diagnosed with lung cancer (Yes or No).

Methodology

1. **Data Collection and Loading:**
    - Imported the dataset `lung_cancer_data.csv` using Pandas.
    - Created a backup dataframe (`sf`) to preserve the original data.

2. **Data Preprocessing:**
    - Filled missing values in columns with their respective mode values.
    - Dropped non-essential columns like `treatment_type` and `cancer_stage`.
    - Applied label encoding to convert categorical data into numerical format.
    - Normalized numerical values using `StandardScaler()` for consistency across features.

3. **Exploratory Data Analysis (EDA):**
    - Utilized functions like `.info()`, `.isnull()`, `.sum()`, and `.describe()` to summarize the data.
    - Created visualizations such as bar graphs, histograms, box plots, and line plots to identify patterns.
    - Noted key observations, including the distribution of game duration and a comparison of gold earned by winning versus losing teams.

4. **Model Implementation:**
    - Split the dataset into 80% training and 20% testing using `train_test_split()`.
    - Applied machine learning models:
        - Logistic Regression
        - K-Nearest Neighbors (KNN)
        - Support Vector Machine (SVM)
        - Decision Tree
        - Random Forest
        - Naive Bayes

5. **Performance Evaluation:**
    - Assessed models using accuracy scores, confusion matrices, and classification reports.
    - Found that Logistic Regression and Supporting Vector Classifiers performed best.
    - Random Forest also showed strong results by leveraging its ability to handle complex decision boundaries.

Code Walkthrough

Importing Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

Reading the dataset

```
df = pd.read_csv('lung_cancer_prediction_dataset.csv')
sf = pd.read_csv('lung_cancer_prediction_dataset.csv')
```

Data Understanding and Inspection

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220632 entries, 0 to 220631
Data columns (total 22 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   ID                          220632 non-null  int64
 1   Country                     220632 non-null  object
 2   Population_Size             220632 non-null  int64
 3   Age                         220632 non-null  int64
 4   Gender                      220632 non-null  object
 5   Smoker                      220632 non-null  object
 6   Years_of_Smoking            220632 non-null  int64
 7   Cigarettes_per_Day          220632 non-null  int64
 8   Passive_Smoker              220632 non-null  object
 9   Family_History              220632 non-null  object
 10  Lung_Cancer_Diagnosis       220632 non-null  object
 11  Survival_Years              220632 non-null  int64
 12  Adenocarcinoma_Type         220632 non-null  object
 13  Air_Pollution_Exposure      220632 non-null  object
 14  Occupational_Exposure       220632 non-null  object
 15  Indoor_Pollution            220632 non-null  object
 16  Healthcare_Access           220632 non-null  object
 17  Early_Detection             220632 non-null  object
 18  Developed_or_Developing     220632 non-null  object
 19  Annual_Lung_Cancer_Deaths   220632 non-null  int64
 20  Lung_Cancer_Prevalence_Rate 220632 non-null  float64
 21  Mortality_Rate              220632 non-null  float64
dtypes: float64(2), int64(7), object(13)
memory usage: 37.0+ MB
```

```
df.isnull().sum()
```

```
ID                                0
Country                           0
Population_Size                   0
Age                               0
Gender                            0
Smoker                            0
Years_of_Smoking                  0
Cigarettes_per_Day                0
Passive_Smoker                    0
Family_History                    0
Lung_Cancer_Diagnosis             0
Cancer_Stage                 211671
Survival_Years                    0
Adenocarcinoma_Type               0
Air_Pollution_Exposure            0
Occupational_Exposure             0
Indoor_Pollution                  0
Healthcare_Access                 0
Early_Detection                   0
Treatment_Type               213968
Developed_or_Developing           0
Annual_Lung_Cancer_Deaths         0
Lung_Cancer_Prevalence_Rate       0
Mortality_Rate                    0
dtype: int64
```

Dropping columns with 75% or higher null values

```
df.isnull().mean() * 100
```

```
ID                           0.000000
Country                      0.000000
Population_Size              0.000000
Age                          0.000000
Gender                       0.000000
Smoker                       0.000000
Years_of_Smoking             0.000000
Cigarettes_per_Day           0.000000
Passive_Smoker               0.000000
Family_History               0.000000
Lung_Cancer_Diagnosis        0.000000
Cancer_Stage                95.938486
Survival_Years               0.000000
Adenocarcinoma_Type          0.000000
Air_Pollution_Exposure       0.000000
Occupational_Exposure        0.000000
Indoor_Pollution             0.000000
Healthcare_Access            0.000000
Early_Detection              0.000000
Treatment_Type              96.979586
Developed_or_Developing      0.000000
Annual_Lung_Cancer_Deaths    0.000000
Lung_Cancer_Prevalence_Rate  0.000000
Mortality_Rate               0.000000
```

```
df.drop(columns=['Cancer_Stage'],inplace=True)
df.drop(columns=['Treatment_Type'], inplace=True)

df.info()
```

```
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   ID                         220632 non-null  int64
 1   Country                    220632 non-null  object
 2   Population_Size            220632 non-null  int64
 3   Age                        220632 non-null  int64
 4   Gender                     220632 non-null  object
 5   Smoker                     220632 non-null  object
 6   Years_of_Smoking           220632 non-null  int64
 7   Cigarettes_per_Day         220632 non-null  int64
 8   Passive_Smoker             220632 non-null  object
 9   Family_History             220632 non-null  object
 10  Lung_Cancer_Diagnosis      220632 non-null  object
 11  Survival_Years             220632 non-null  int64
 12  Adenocarcinoma_Type        220632 non-null  object
 13  Air_Pollution_Exposure     220632 non-null  object
 14  Occupational_Exposure      220632 non-null  object
 15  Indoor_Pollution           220632 non-null  object
 16  Healthcare_Access          220632 non-null  object
 17  Early_Detection            220632 non-null  object
 18  Developed_or_Developing    220632 non-null  object
 19  Annual_Lung_Cancer_Deaths  220632 non-null  int64
 20  Lung_Cancer_Prevalence_Rate  220632 non-null  float64
 21  Mortality_Rate             220632 non-null  float64
```

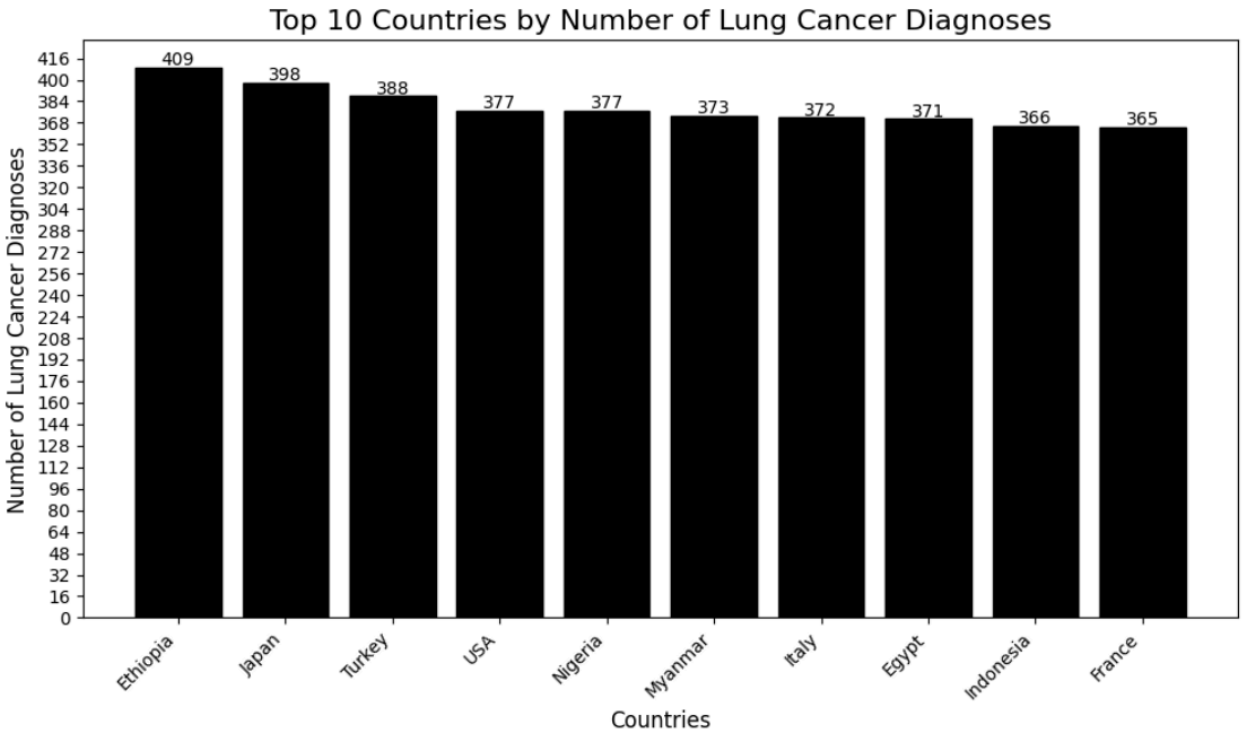
Filling the missing values

```
# 404 not found
# We didn't find any missing values in the rest of the columns
```

Visualization

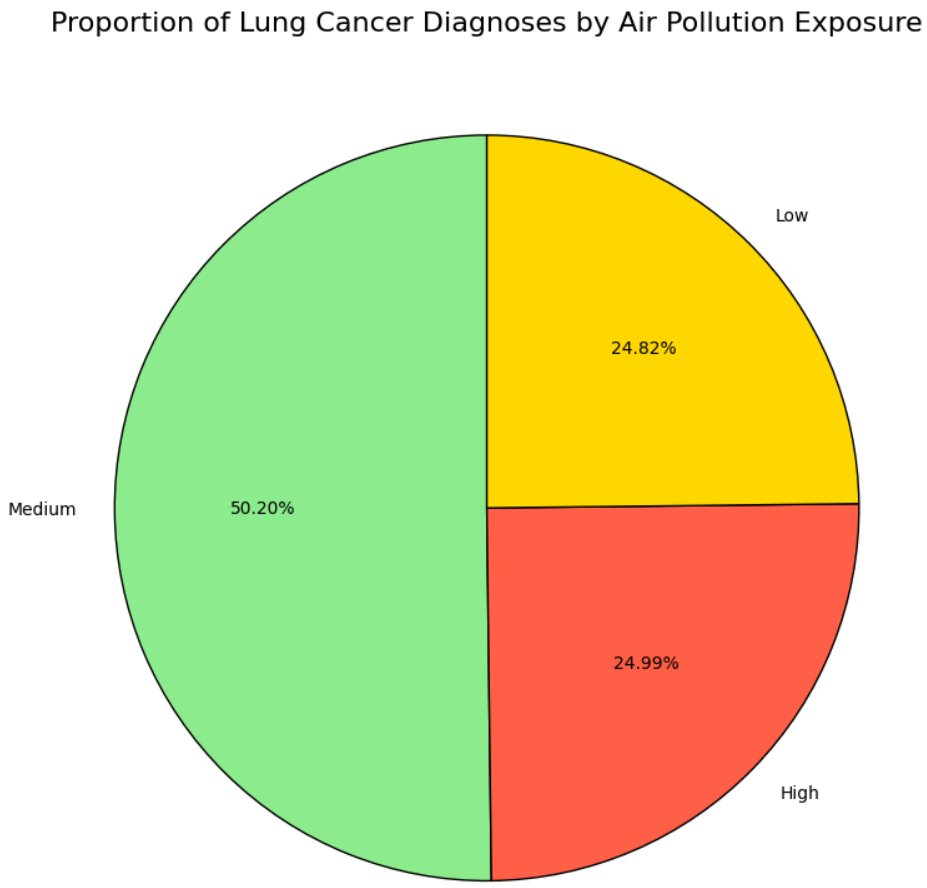
I. Bar Graph

```
Lung_cancer_counts=df[df['Lung_Cancer_Diagnosis']=='Yes']['Country'].value_counts()
sorted_countries = lung_cancer_counts.sort_values(ascending=False)
top_n = 10 # we want the top 10 countries
top_countries = sorted_countries.head(top_n)
plt.figure(figsize=(10, 6))
bars = plt.bar(top_countries.index, top_countries.values, color='black',
edgecolor='black')
plt.yticks(range(0, y_max + y_step, y_step), fontsize=10)
plt.title('Top 10 Countries by Number of Lung Cancer Diagnoses', fontsize=16)
plt.xlabel('Countries', fontsize=12)
plt.ylabel('Number of Lung Cancer Diagnoses', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.tight_layout()
plt.show()
```



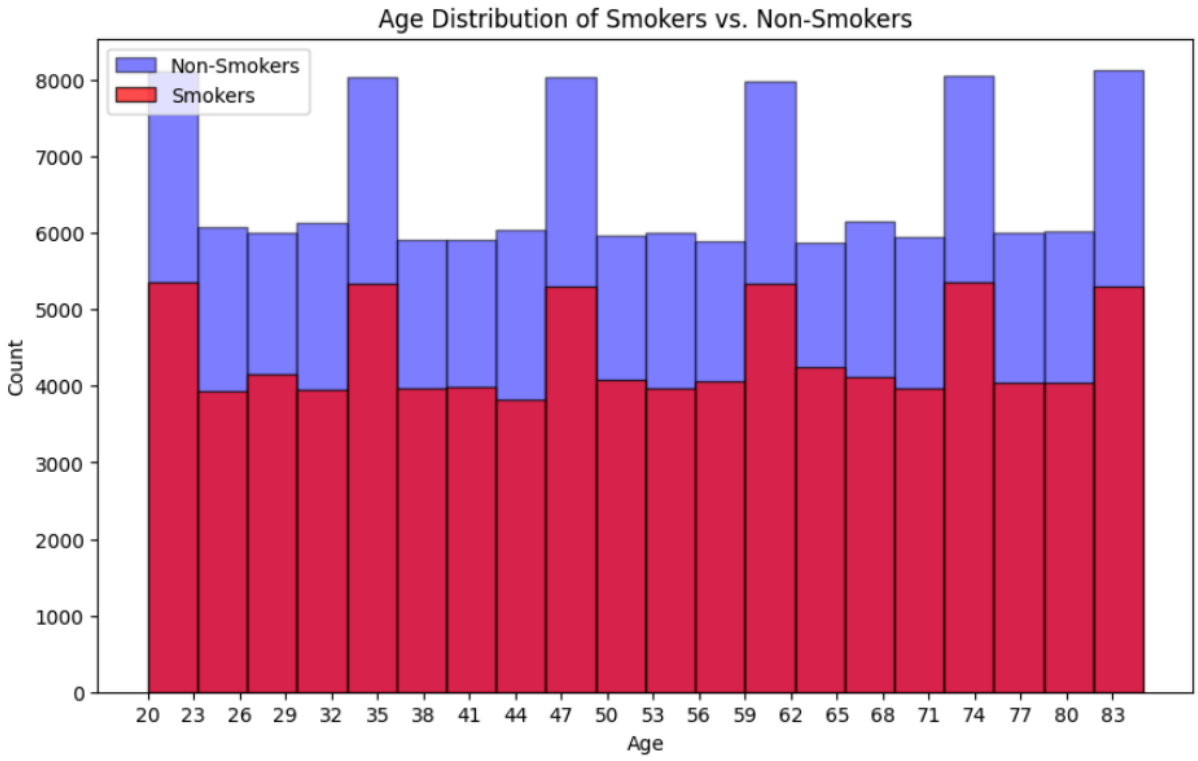
II. Pie Chart

```
lung_cancer_cases = df[df['Lung_Cancer_Diagnosis'] == 'Yes']
pollution_exposure_counts = lung_cancer_cases['Air_Pollution_Exposure'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(
    pollution_exposure_counts.values,
    labels=pollution_exposure_counts.index,
    autopct='%1.2f%%',
    startangle=90,
    colors=['lightgreen', 'tomato', 'gold'],
    wedgeprops={'edgecolor': 'black'}
)
plt.title('Proportion of Lung Cancer Diagnoses by Air Pollution Exposure',
fontsize=16)
plt.tight_layout()
plt.show()
```



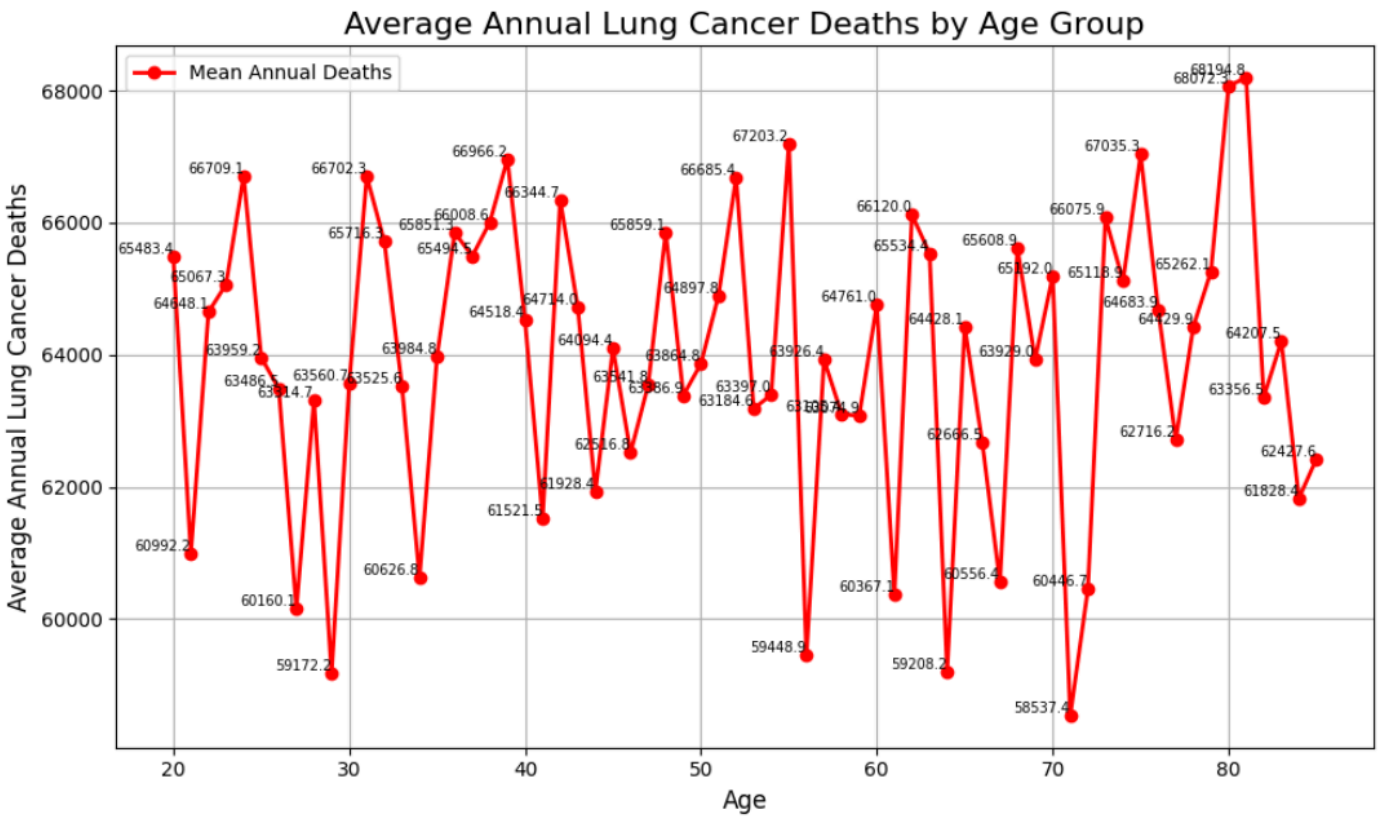
III. Histogram

```
smokers = df[df["Smoker"] == "Yes"]["Age"]
non_smokers = df[df["Smoker"] == "No"]["Age"]

plt.figure(figsize=(10, 6))
plt.hist(non_smokers, bins=20, alpha=0.5, label="Non-Smokers", color="blue",
edgecolor="black")
plt.hist(smokers, bins=20, alpha=0.7, label="Smokers", color="red",
edgecolor="black")

plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Age Distribution of Smokers vs. Non-Smokers")
plt.legend()
plt.show()
```



IV. Line Plot

```
ge_grouped = df.groupby('Age')['Annual_Lung_Cancer_Deaths'].mean()
plt.figure(figsize=(10, 6))
plt.plot(
    age_grouped.index,
    age_grouped.values,
    marker='o',
    color='red',
    linestyle='-',
    linewidth=2,
    label='Mean Annual Deaths'
)
plt.title('Average Annual Lung Cancer Deaths by Age Group', fontsize=16)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Average Annual Lung Cancer Deaths', fontsize=12)
plt.grid(True)
plt.legend(fontsize=10)
plt.tight_layout()
plt.show()
```



Finding Insights by Implementing Machine Learning Algorithms

Defining Feature Map

```
X = data[['Age', 'Smoker', 'Years_of_Smoking', 'Cigarettes_per_Day',
          'Passive_Smoker', 'Air_Pollution_Exposure',
          'Occupational_Exposure', 'Early_Detection',
          'Developed_or_Developing', 'Family_History',
          'Indoor_Pollution', 'Healthcare_Access']]
```

Defining Target Variable

```
target = df['Lung_Cancer_Diagnosis'].map({'No': 0, 'Yes': 1})
```

Encoding Categorical Variables

```
for column in X.columns:
    if X[column].dtype == 'object':
        le = LabelEncoder()
        X[column] = le.fit_transform(X[column])
```

Splitting the Dataset into Training and Testing sets

```
X_train, X_test, y_train, y_test = train_test_split(features_encoded,
target_encoded, test_size=0.2, random_state=42)
```

Standardizing the Features

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logistic Regression

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log_reg))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log_reg))
print("Classification Report:\n", classification_report(y_test,
y_pred_log_reg, zero_division = 0))
```

```
Logistic Regression Accuracy: 0.9593899426654883
Confusion Matrix:
 [[42335     0]
 [ 1792     0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     42335
           1       0.00      0.00      0.00      1792

    accuracy                           0.96     44127
   macro avg       0.48      0.50      0.49     44127
weighted avg       0.92      0.96      0.94     44127
```

K Nearest Neighbours

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
print("Classification Report:\n", classification_report(y_test, y_pred_knn))
```

```
KNN Accuracy: 0.9583928207220069
Confusion Matrix:
 [[42288    47]
 [ 1789     3]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     42335
           1       0.06      0.00      0.00      1792

    accuracy                           0.96     44127
   macro avg       0.51      0.50      0.49     44127
weighted avg       0.92      0.96      0.94     44127
```

Support Vector Machine

```
svc_model = SVC(kernel = 'linear')
svc_model.fit(X_train, y_train)
y_pred_svc = svc_model.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svc))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svc))
print("Classification Report:\n", classification_report(y_test, y_pred_svc))
```

```
SVM Accuracy: 0.9593899426654883
Confusion Matrix:
 [[42335     0]
 [ 1792     0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     42335
           1       0.00      0.00      0.00      1792

    accuracy                           0.96     44127
   macro avg       0.48      0.50      0.49     44127
weighted avg       0.92      0.96      0.94     44127
```

Decision Tree

```
DC_model = DecisionTreeClassifier(criterion = 'gini', random_state = 42)
DC_model.fit(X_train, y_train)
y_pred = DC_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
class_report = classification_report(y_test, y_pred, zero_division = 0)
print('accuracy:', accuracy, '\n')
print('classification report: \n', class_report, '\n')
```

```
Decision Tree Accuracy: 0.9284111768305119

Classification report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     42335
           1       0.08      0.07      0.07      1792

    accuracy                           0.93     44127
   macro avg       0.52      0.52      0.52     44127
weighted avg       0.92      0.93      0.93     44127
```

Random Forest

```
RF_model = RandomForestClassifier(n_estimators = 100, criterion = 'gini',
random_state = 42)
RF_model.fit(X_train, y_train)
y_pred_rf = RF_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
class_report_rf = classification_report(y_test, y_pred_rf, zero_division=0)
print(f"\nRandom Forest Accuracy: {accuracy_rf}")
print("Random Forest Classification Report:\n", class_report_rf)
```

```
Random Forest Accuracy: 0.9576223174020441
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     42335
           1       0.07      0.00      0.01      1792

    accuracy                           0.96     44127
   macro avg       0.51      0.50      0.49     44127
weighted avg       0.92      0.96      0.94     44127
```

Naive Bayes

```
NB_model = GaussianNB()
NB_model.fit(X_train, y_train)
y_pred_nb = NB_model.predict(X_test)
print(f'Naive Bayes Accuracy: {accuracy_score(y_test, y_pred_nb)}')
print(f'Classification Report: \n{classification_report(y_test, y_pred_nb)}')
```

```
Naive Bayes Accuracy: 0.9555147642033223
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     42335
           1       0.07      0.01      0.01      1792

    accuracy                           0.96     44127
   macro avg       0.51      0.50      0.50     44127
weighted avg       0.92      0.96      0.94     44127
```

Conclusion

This project provided valuable insights into lung cancer risk factors and developed predictive models to aid early detection. Key findings include:

- Age and smoking habits strongly influence lung cancer prevalence.
- Environmental factors like air pollution significantly contribute to lung cancer risk.

This project demonstrates the power of data science in addressing real-world health challenges. By leveraging EDA and machine learning, we can make meaningful contributions to medical research and public health initiatives.