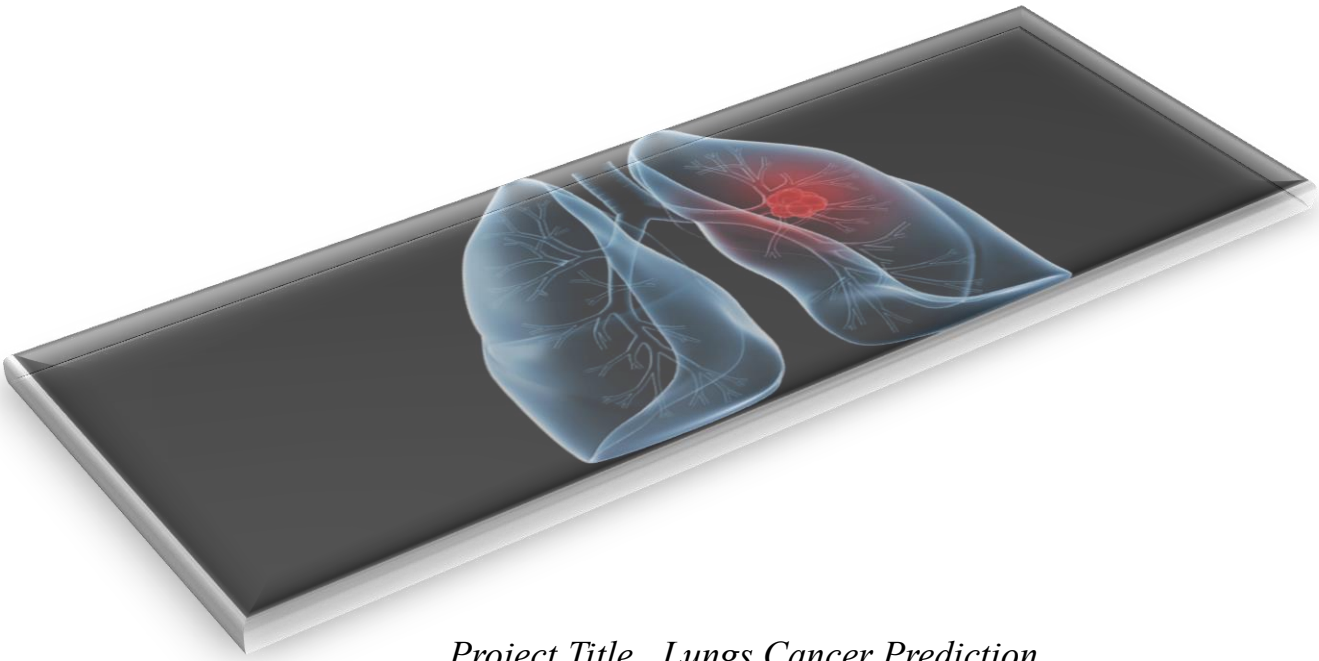
Machine Learning- Naan Mudhalvan

Annai Mira College of Engineering and Technology



Project Title	*Lungs Cancer Prediction*
Reg. No	513520106001
Name	Arishwaran G
Degree	BE ECE
Sem	07

Predicting lungs cancer using machine learning involves several steps, including preprocess the data, extracting relevant features, and training a model:-

1. Data Collection:

Objective: Obtain a dataset that includes information about patients, such as demographics, medical history, and any relevant features that might contribute to the prediction of lung cancer.

Actions:

1. Identify reliable sources for medical data, such as hospitals, research institutions, or publicly available datasets.
2. Ensure that the dataset includes features that are potentially relevant to lung cancer prediction, such as age, gender, smoking history, and other medical indicators.
3. Obtain labeled instances indicating whether each patient has lung cancer or not.

2. Data Preprocessing:

Objective: Prepare the dataset for machine learning by handling missing data, encoding categorical variables, normalizing/standardizing numerical features, and addressing outliers.

Actions:

1. Handle missing data by imputing values or removing instances with missing data.
2. Convert categorical variables into a numerical format, e.g., using one-hot encoding.
3. Normalize or standardize numerical features to bring them to a common scale.
4. Identify and handle outliers to prevent them from disproportionately influencing the model.

3. Feature Extraction:

Objective: Extract relevant features from the data that might contribute to the prediction of lung cancer.

Actions:

1. Identify features that are likely to be important for predicting lung cancer based on domain knowledge or exploratory data analysis.
2. Extract features such as demographic information, smoking history, environmental exposures, genetic factors, and relevant medical history.

4. Feature Selection:

Objective: Select the most important features to improve model performance and reduce overfitting.

Actions:

1. Use techniques like Recursive Feature Elimination (RFE) to recursively remove less important features.
2. Utilize feature importance scores from tree-based models like Random Forest.
3. Perform dimensionality reduction techniques like Principal Component Analysis (PCA) if the dataset has a large number of features.

5. Model Selection:

Objective: Choose a suitable machine learning algorithm for the lung cancer prediction task.

Actions:

1. Consider the characteristics of your data and the nature of the problem (binary classification in this case).
2. Common algorithms for binary classification include Logistic Regression, Support Vector Machines (SVM), Random Forest, and Gradient Boosting algorithms (e.g., XGBoost).

6. Model Training:

Objective: Train the selected machine learning model using the training dataset.

Actions:

1. Split the dataset into training and testing sets to evaluate the model's performance.
2. Feed the training data into the chosen algorithm and adjust its parameters for optimal performance.

7. Model Evaluation:

Objective: Assess the model's performance on the testing dataset using relevant evaluation metrics.

Actions:

1. Evaluate the model using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.
2. Fine-tune hyperparameters to optimize the model's performance.

8. Cross-Validation:

Objective: Perform cross-validation to assess the model's generalization performance.

Actions:

1. Use techniques like k-fold cross-validation to evaluate the model's performance across different subsets of the data.
2. Ensure that the model is robust and not overfitting to the specific training dataset.

9. Interpretability:

Objective: Understand the factors contributing to predictions and make the model more interpretable.

Actions:

1. Depending on the chosen model, interpret feature importance scores.
2. Use model-agnostic techniques like SHAP (SHapley Additive exPlanations) values to understand individual predictions.

10. Deployment:

Objective: Deploy the model for real-world use.

Actions:

1. Deploy the model in a production environment where it can be integrated into a healthcare system or used by relevant stakeholders.
2. Monitor the model's performance over time and update it as needed.

‘Lungs Cancer Prediction using Machine Learning – Python Program’

#IMPORT PACKAGES

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from tqdm.notebook import tqdm
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
```

#PREPROCESSING

```
df = pd.read_csv('lung-cancer_csv.csv')
```

```
df.shape
```

```
df.info()
```

#FEATURE EXTRACTION

```
df.describe()
```

```
columns = list(df.columns)
for col in columns:
        if col == 'class':
                continue

        filtered_columns = [col]
        for col1 in df.columns:
                if((col == col1) | (col == 'class')):
                        continue

                val = df[col].corr(df[col1])

                if val > 0.7:
                        # If the correlation between the two
                        # features is more than 0.7 remove
                        columns.remove(col1)
                        continue
                else:
                        filtered_columns.append(col1)

        # After each iteration filter out the columns
        # which are not highly correlated features.
        df = df[filtered_columns]
df.shape
```

#TRAINING MODEL

```
x = df['class'].value_counts()
plt.pie(x.values,
                labels = x.index,
                autopct='%1.1f%%')
plt.show()
```

```
features = df.drop('class', axis=1)
target = df['class']


X_train, X_val,\
        Y_train, Y_val = train_test_split(features, target, test_size=0.2,
                                                                random_state=10)
X_train.shape, X_val.shape
```

OUTPUT

ARISHWARAN G.ipynb | lung-cancer_csv.csv

Code | Python (Pyodide)

[33]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from tqdm.notebook import tqdm
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
```

[34]:
```
df = pd.read_csv('lung-cancer_csv.csv')
```

[35]:
```
df.shape
```

[35]: (32, 57)

[36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 57 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   class        32 non-null     int64
 1   attribute2   32 non-null     int64
 2   attribute3   32 non-null     int64
 3   attribute4   32 non-null     int64
 4   attribute5   28 non-null     float64
 5   attribute6   32 non-null     int64
 6   attribute7   32 non-null     int64
 7   attribute8   32 non-null     int64
 8   attribute9   32 non-null     int64
 9   attribute10  32 non-null     int64
 10  attribute11  32 non-null     int64
 11  attribute12  32 non-null     int64
 12  attribute13  32 non-null     int64
 13  attribute14  32 non-null     int64
 14  attribute15  32 non-null     int64
 15  attribute16  32 non-null     int64
 16  attribute17  32 non-null     int64
 17  attribute18  32 non-null     int64
 18  attribute19  32 non-null     int64
 19  attribute20  32 non-null     int64
 20  attribute21  32 non-null     int64
 21  attribute22  32 non-null     int64
 22  attribute23  32 non-null     int64
 23  attribute24  32 non-null     int64
 24  attribute25  32 non-null     int64
 25  attribute26  32 non-null     int64
 26  attribute27  32 non-null     int64
 27  attribute28  32 non-null     int64
 28  attribute29  32 non-null     int64
 29  attribute30  32 non-null     int64
 30  attribute31  32 non-null     int64
 31  attribute32  32 non-null     int64
 32  attribute33  32 non-null     int64
 33  attribute34  32 non-null     int64
 34  attribute35  32 non-null     int64
 35  attribute36  32 non-null     int64
 36  attribute37  32 non-null     int64
 37  attribute38  32 non-null     int64
 38  attribute39  31 non-null     float64
 39  attribute40  32 non-null     int64
 40  attribute41  32 non-null     int64
 41  attribute42  32 non-null     int64
 42  attribute43  32 non-null     int64
 43  attribute44  32 non-null     int64
 44  attribute45  32 non-null     int64
 45  attribute46  32 non-null     int64
 46  attribute47  32 non-null     int64
 47  attribute48  32 non-null     int64
 48  attribute49  32 non-null     int64
 49  attribute50  32 non-null     int64
 50  attribute51  32 non-null     int64
 51  attribute52  32 non-null     int64
 52  attribute53  32 non-null     int64
 53  attribute54  32 non-null     int64
 54  attribute55  32 non-null     int64
 55  attribute56  32 non-null     int64
 56  attribute57  32 non-null     int64
dtypes: float64(2), int64(55)
memory usage: 14.3 KB
```

[37]:
```
df.describe()
```

[37]:

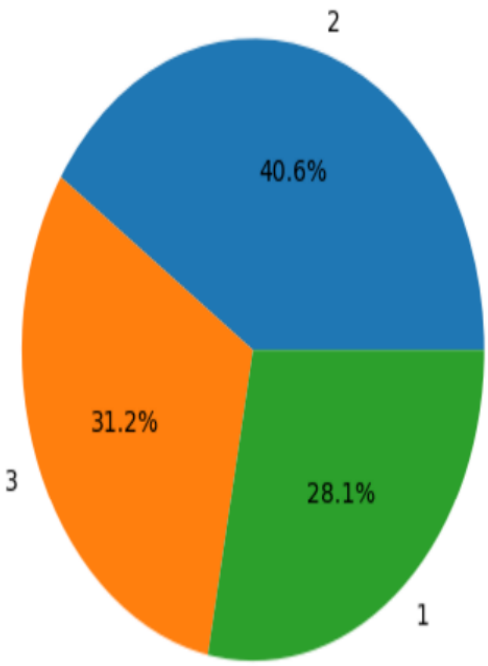
	class	attribute2	attribute3	attribute4	attribute5	attribute6	attribute7	attribute8	attribute9	attribute10	...
count	32.000000	32.000000	32.000000	32.000000	28.000000	32.000000	32.000000	32.000000	32.000000	32.000000	...
mean	2.031250	0.031250	2.375000	2.031250	1.392857	0.281250	2.187500	2.125000	2.406250	1.156250	...
std	0.782237	0.176777	0.553581	1.031265	0.566947	0.456803	0.737804	0.751343	0.756024	0.514899	...
min	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	...
25%	1.000000	0.000000	2.000000	1.750000	1.000000	0.000000	2.000000	2.000000	2.000000	1.000000	...
50%	2.000000	0.000000	2.000000	2.000000	1.000000	0.000000	2.000000	2.000000	3.000000	1.000000	...
75%	3.000000	0.000000	3.000000	3.000000	2.000000	1.000000	3.000000	3.000000	3.000000	1.000000	...
max	3.000000	1.000000	3.000000	3.000000	2.000000	1.000000	3.000000	3.000000	3.000000	3.000000	...

8 rows × 57 columns

[39]:
```
columns = list(df.columns)
for col in columns:
    if col == 'class':
        continue

    filtered_columns = [col]
    for col1 in df.columns:
        if((col == col1) | (col == 'class')):
            continue

        val = df[col].corr(df[col1])

        if val > 0.7:
            # If the correlation between the two
            # features is more than 0.7 remove
            columns.remove(col1)
            continue
        else:
            filtered_columns.append(col1)

    # After each iteration filter out the columns
    # which are not highly correlated features.
    df = df[filtered_columns]
df.shape
```

[39]: (32, 53)

[55]:
```
x = df['class'].value_counts()
plt.pie(x.values,
        labels = x.index,
        autopct='%1.1f%%')
plt.show()
```



[52]:
```
features = df.drop('class', axis=1)
target = df['class']

X_train, X_val,\
    Y_train, Y_val = train_test_split(features, target,
                                      test_size=0.2,
                                      random_state=10)
X_train.shape, X_val.shape
```

[52]: ((25, 52), (7, 52))