

lungcancerprediction

November 28, 2024

```
[2]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, \
    confusion_matrix
```

```
[3]: df = pd.read_csv('/content/drive/MyDrive/survey lung cancer.csv')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   GENDER                309 non-null   object
 1   AGE                  309 non-null   int64
 2   SMOKING              309 non-null   int64
 3   YELLOW_FINGERS       309 non-null   int64
 4   ANXIETY              309 non-null   int64
 5   PEER_PRESSURE        309 non-null   int64
 6   CHRONIC DISEASE      309 non-null   int64
 7   FATIGUE              309 non-null   int64
 8   ALLERGY              309 non-null   int64
 9   WHEEZING             309 non-null   int64
10   ALCOHOL CONSUMING    309 non-null   int64
11   COUGHING             309 non-null   int64
12   SHORTNESS OF BREATH  309 non-null   int64
13   SWALLOWING DIFFICULTY 309 non-null   int64
14   CHEST PAIN           309 non-null   int64
15   LUNG_CANCER          309 non-null   object
dtypes: int64(14), object(2)
memory usage: 38.8+ KB
```

```
[5]: df.head()
```

```
[5]:
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	\
0	M	69	1	2	2	1	
1	M	74	2	1	1	1	
2	F	59	1	1	1	2	
3	M	63	2	2	2	1	
4	F	63	1	2	1	1	

	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	\
0		1	2	1		2	2
1		2	2	2		1	1
2		1	2	1		1	2
3		1	1	1		2	1
4		1	1	1		1	2

	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER	
0		2	2	2	YES
1		2	2	2	YES
2		2	1	2	NO
3		1	2	2	NO
4		2	1	1	NO

```
[6]: # Perform one-hot encoding on the 'gender' column
# Encode the 'GENDER' column: Male ('M') as 1 and Female ('F') as 0
df['GENDER'] = df['GENDER'].apply(lambda x: 1 if x == 'M' else 0)
```

```
[7]: df.head()
```

```
[7]:
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	\
0	1	69	1	2	2	1	
1	1	74	2	1	1	1	
2	0	59	1	1	1	2	
3	1	63	2	2	2	1	
4	0	63	1	2	1	1	

	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	\
0		1	2	1		2	2
1		2	2	2		1	1
2		1	2	1		1	2
3		1	1	1		2	1
4		1	1	1		1	2

	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER	
0		2	2	2	YES
1		2	2	2	YES
2		2	1	2	NO
3		1	2	2	NO
4		2	1	1	NO

```
[8]: # check for duplicate records
df.duplicated().sum()
```

```
[8]: 33
```

```
[9]: # Drop duplicates records
df.drop_duplicates(inplace=True)
```

```
[10]: df.duplicated().sum()
```

```
[10]: 0
```

```
[12]: # check for null values
df.isna().sum()
```

```
[12]: 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
```

Logistic Regression Model

```
[13]: # Initialize the logistic regression model
logistic_model = LogisticRegression(max_iter=1000, random_state=42)
```

```
[14]: # Feature Selection and Engineering
# Identifying features (X) and target variable (y)
X = df.drop(columns=['LUNG_CANCER'], axis=1)
y = df['LUNG_CANCER']
```

```
[15]: # Train-Test Split
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=42)
```

```
[16]: # Train the model on the training data
logistic_model.fit(X_train, y_train)
```

```
[16]: LogisticRegression(max_iter=1000, random_state=42)
```

```
[17]: # Make predictions on the test set
y_pred = logistic_model.predict(X_test)
```

```
[18]: # Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.891566265060241
```

```
[19]: # Confusion matrix
confusion_matrix(y_test, y_pred)
```

```
[19]: array([[ 4,  9],
        [ 0, 70]])
```

```
[20]: # Check accuracy of model
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
NO	1.00	0.31	0.47	13
YES	0.89	1.00	0.94	70
accuracy			0.89	83
macro avg	0.94	0.65	0.71	83
weighted avg	0.90	0.89	0.87	83

Random Forest Classifier

```
[21]: from sklearn.ensemble import RandomForestClassifier
```

```
[22]: # Initialize the random forest classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
[23]: # Train the model on the training data
rf_model.fit(X_train, y_train)
```

```
[23]: RandomForestClassifier(random_state=42)
```

```
[24]: # Make predictions on the test set
y_pred_rf = rf_model.predict(X_test)
```

```
[33]: # Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred_rf)
print(accuracy)
```

0.891566265060241

```
[25]: # Confusion matrix
confusion_matrix(y_test, y_pred_rf)
```

```
[25]: array([[ 4,  9],
          [ 0, 70]])
```

```
[26]: # Check accuracy of model
print(classification_report(y_test,y_pred_rf))
```

	precision	recall	f1-score	support
NO	1.00	0.31	0.47	13
YES	0.89	1.00	0.94	70
accuracy			0.89	83
macro avg	0.94	0.65	0.71	83
weighted avg	0.90	0.89	0.87	83

Support Vector Machine model

```
[27]: from sklearn.svm import SVC
```

```
[28]: # Initialize the Support Vector Machine model
svm_model = SVC(kernel='linear', random_state=42)
```

```
[29]: # Train the model on the training data
svm_model.fit(X_train, y_train)
```

```
[29]: SVC(kernel='linear', random_state=42)
```

```
[30]: # Make predictions on the test set
y_pred_svm = svm_model.predict(X_test)
```

```
[34]: # Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred_svm)
print(accuracy)
```

0.9397590361445783

```
[31]: # Confusion matrix
confusion_matrix(y_test, y_pred_svm)
```

```
[31]: array([[ 8,  5],  
           [ 0, 70]])
```

```
[32]: # Check accuracy of model  
print(classification_report(y_test,y_pred_svm))
```

	precision	recall	f1-score	support
NO	1.00	0.62	0.76	13
YES	0.93	1.00	0.97	70
accuracy			0.94	83
macro avg	0.97	0.81	0.86	83
weighted avg	0.94	0.94	0.93	83

Model Comparison Report

Logistic regression model and Random Classifier have have same accuracy of 89%. Whereas Support Vector Machine model has accuracy of 94%. Therefore, Support Vector Machine model is best model for production.