Lung Cancer Prediction Model

Group 20

Mahendra Rasuri

Shriya Kenkre

(857)-707-8014

(847)-530-3400

rasuri.m@northeastern.edu

kenkre.s@northeastern.edu

Percentage of efforts contributed by Student 1: 50%

Percentage of efforts contributed by Student 2: 50%

Signature of Student 1: Mahendra Rasuri

Signature of Student 2: Shriya Kenkre

Submission Date: 05/1/2022

Problem Setting: The success of a cancer prediction system allows people to learn about their cancer risk at a cheap cost and to make informed decisions depending on their cancer risk status. We'll investigate the connection and develop a predictive model for lung cancer diagnosis. We'll do so by modelling traits that are/could be linked to the condition.

Project Overview: Lung cancer is the world's largest cause of cancer death. The current study examines the relationship between lung cancer screening results and later lung cancer risk after controlling for other significant lung cancer risk factors. We have worked on a model to predict the risk of a person having lung cancer.

Problem Definition: To determine whether lung cancer has a disease phenotype. People with lung cancer will have a disease phenotype, while those without will not, which the model will be able to detect using the data in the features.

Data Sources: The data is collected from the website online lung cancer prediction system.

1. https://www.kaggle.com/

Data Description:

Total no. of attributes:16

No .of instances:284

Attribute information:

1. Gender: M(male), F(female)

2. Age: Age of the patient

3. Smoking: YES=2 , NO=1.

4. Yellow fingers: YES=2 , NO=1.

5. Anxiety: YES=2 , NO=1.

6. Peer_pressure: YES=2 , NO=1.

7. Chronic Disease: YES=2 , NO=1.

8. Fatigue: YES=2 , NO=1.

9. Allergy: YES=2 , NO=1.

10. Wheezing: YES=2 , NO=1.

11. Alcohol: YES=2 , NO=1.

12. Coughing: YES=2 , NO=1.

13. Shortness of Breath: YES=2 , NO=1.

14. Swallowing Difficulty: YES=2 , NO=1.

15. Chest pain: YES=2 , NO=1.

16. Lung Cancer: YES , NO.

Data Processing:

In this step we have split the main csv file two dataframes namely p and q. In dataframe p we have included all the features and in q we have taken the deciding variable “Lung_cancer”, As we cannot work with characters M & F (which is gender with respect to male and female), they are changed to numeric values of 1 and 0 (mapped using map function) respectively and the column “Gender” is dropped as the new mapped values come under the newly created column “Sex”.

Code Snapshot:

In [5]:
```
list_of_features = ['GENDER','AGE','SMOKING','YELLOW_FINGERS','ANXIETY','PEER_PRESSURE','CHRONIC DISEASE',
                    'FATIGUE ','ALLERGY ','WHEEZING','ALCOHOL CONSUMING','COUGHING','SHORTNESS OF BREATH','SWALLOWING DIFFICU
p = file[list_of_features]
q = file['LUNG_CANCER']
```

In [6]:
```
p['SEX'] = p['GENDER'].map({'M':1, 'F':0})
p_updated = p.drop(columns = 'GENDER')
p_updated.head()
```

Out[6]:

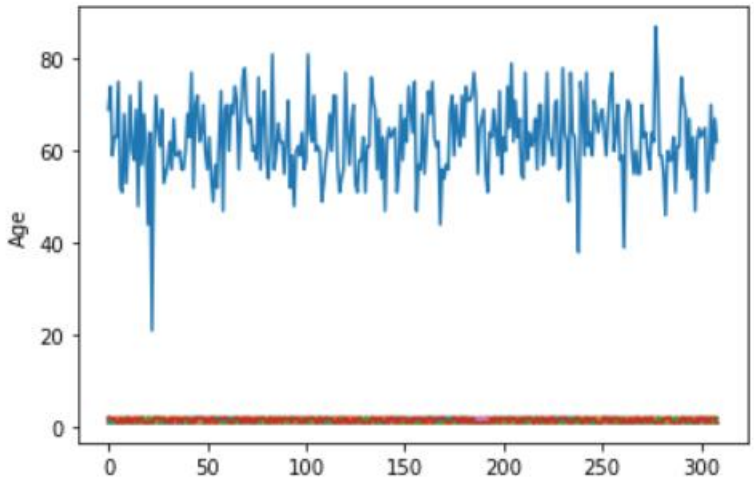
	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH
0	69	1	2	2	1	1	2	1	2	2	2	2
1	74	2	1	1	1	2	2	2	1	1	1	2
2	59	1	1	1	2	1	2	1	2	1	2	2
3	63	2	2	2	1	1	1	1	1	2	1	1
4	63	1	2	1	1	1	1	1	2	1	2	2

Data Visualization:

We have created two plots using “matplotlib” function.

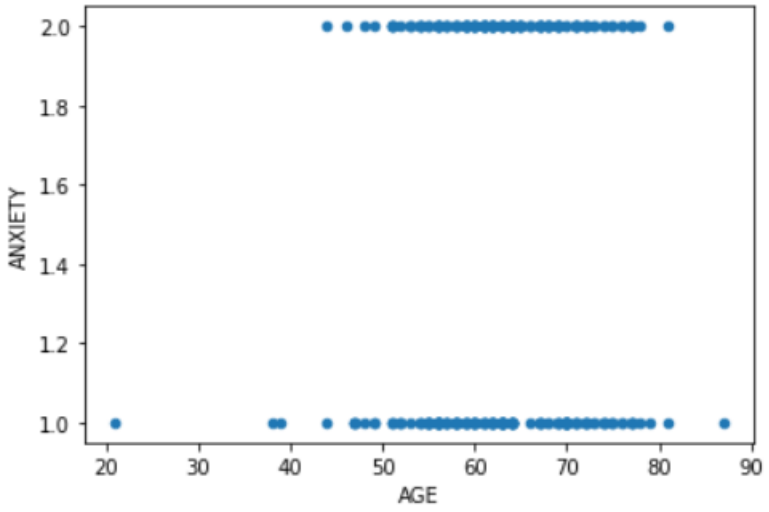
- The first one is a plot between the age of a person and if the person has cancer or not. The reason behind choosing to make this plot is to find if there is any correlation between age and lung cancer. Though we came to a conclusion that as a person’s age increases from 40s significantly the chances of getting lung cancer increases, the disease also doesn’t spare the ones younger than that, but the significance is much lesser.

```
age = file.plot()
plt.ylabel('Age')
age.get_legend().remove()
```



- The second plot which is a scatter plot involves the factor of correlation between anxiety and age. From the graph plotted, it can be concluded that there is no correlation between anxiety and age. It can attack someone in their twenties as well as someone in their sixties.

```
In [8]: file.plot.scatter(x="AGE", y="ANXIETY")
Out[8]: <AxesSubplot:xlabel='AGE', ylabel='ANXIETY'>
```



Problem Identification:

The problem that we have chosen (Lung Cancer prediction) comes under "Regression Modelling" but not "Classification" as we are only trying to idenitfy if a person has lung cancer or not. We are neither classifyinhg nor dividing the data that we have, rather we are focusing on either if a person has cancer (YES) or not (NO).

Output Formatting:

The output that we get whether a person has lung cancer or not comes in the form of either “YES” or “NO” labels. Those labels have been converted into numeric form which is machine readable using “Label Encoder”.

```
In [9]:
LE = LabelEncoder()
LE = LE.fit(q)
encoded_q = LE.transform(q)
q_final = pd.DataFrame(encoded_q)
q_final.head()
```

Out[9]:

	0
0	1
1	1
2	0
3	0
4	0

Splitting Data:

The data has been split into test and train using default settings which is, 25% test data and 75% train data.

```
P_train, P_valid, q_train, q_valid = train_test_split(p_updated, q_final)
```

Model Selection:

The models that we have considered for our data are:

1.Linear Regression

2.Ridge Regression

3.XGBOOST Regression

For Linear Regression:

```
lr = LinearRegression(copy_X= True, fit_intercept = True, normalize = True)
model_LR= LinearRegression()
lr.fit(P_train, q_train)
lr_pred= lr.predict(P_valid)
```

```
predictions = lr.predict(P_valid)
print("Mean Absolute Error: " + str(mean_absolute_error(predictions, q_valid)))
```

```
Mean Absolute Error: 0.1758010793027988
```

For Ridge Regression:

```
model = Ridge(alpha = 0.01, normalize = True)
model.fit(P_train, q_train)
pred_ridge = model.predict(P_valid)

predictions = model.predict(P_valid)
print("Mean Absolute Error: " + str(mean_absolute_error(predictions, q_valid)))
```

```
Mean Absolute Error: 0.1814710974157904
```

For XGBoost:

```
P_train, P_valid, q_train, q_valid = train_test_split(p_updated, q_final)
model = XGBRegressor()
model.fit(P_train, q_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=12,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

```
from sklearn.metrics import mean_absolute_error

predictions = model.predict(P_valid)
print("Mean Absolute Error: " + str(mean_absolute_error(predictions, q_valid)))
```

```
Mean Absolute Error: 0.14321041898578046
```

From the results that we have obtained from testing all the three models, we can see that XGBOOST Regression gives the least Mean Absolute Error (shows the difference between measured and predicted values). So, we have decided to choose XGBOOST for the further steps that we'll be taking.

Model Implementation:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=12,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

```
predictions = model.predict(P_valid)
print("Mean Absolute Error: " + str(mean_absolute_error(predictions, q_valid)))
```

Hence, we are done with implementation.

Performance Evaluation: As we have seen from calculating “Absolute Mean Error”, XGBoost gives the best results with a minor error of 0.14321041898578046.

Conclusion: Across the United States, lung cancer screening programs are being implemented. The risks and costs of a high number of false-positive screens, as well as excessive radiation exposure, are major issues about lung cancer screening. Eliminating nonbeneficial screens is expected to reduce the harms and improve cost-effectiveness. Accurate lung cancer risk prediction models have been shown to be more sensitive in identifying individuals who get lung cancer. Hence our model is a useful addition to the field of medicine. However, access to more data contributing to causes of lung cancer will provide more accurate results.