Predicting Lung Cancer with Machine Learning

September 23, 2022

```python
import pandas as pd
%config Completer.use_jedi = False
import numpy as np
import seaborn as sns  #data visualization library
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, log_loss
from sklearn.model_selection import train_test_split, KFold
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```python
# Read in the data stored in the file 'survey lung cancer.csv'
df = pd.read_csv('survey lung cancer.csv')
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         2
1                2        2        2         1                  1         1
2                1        2        1         2                  1         2
3                1        1        1         1                  2         1
4                1        1        1         2                  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
```

```python
# replace all M/F with 1/0
df.GENDER.replace(['M', 'F'], [1, 0], inplace=True)
```

```
# replace all YES/NO with 1/0
df.LUNG_CANCER.replace(['YES', 'NO'], [1, 0], inplace=True)
```

1 Data analysis

```
[ ]: print("Number of duplicates: ", df.duplicated().sum())
     df = df.drop_duplicates()
     print("Number of duplcates after drop ", df.duplicated().sum())
```
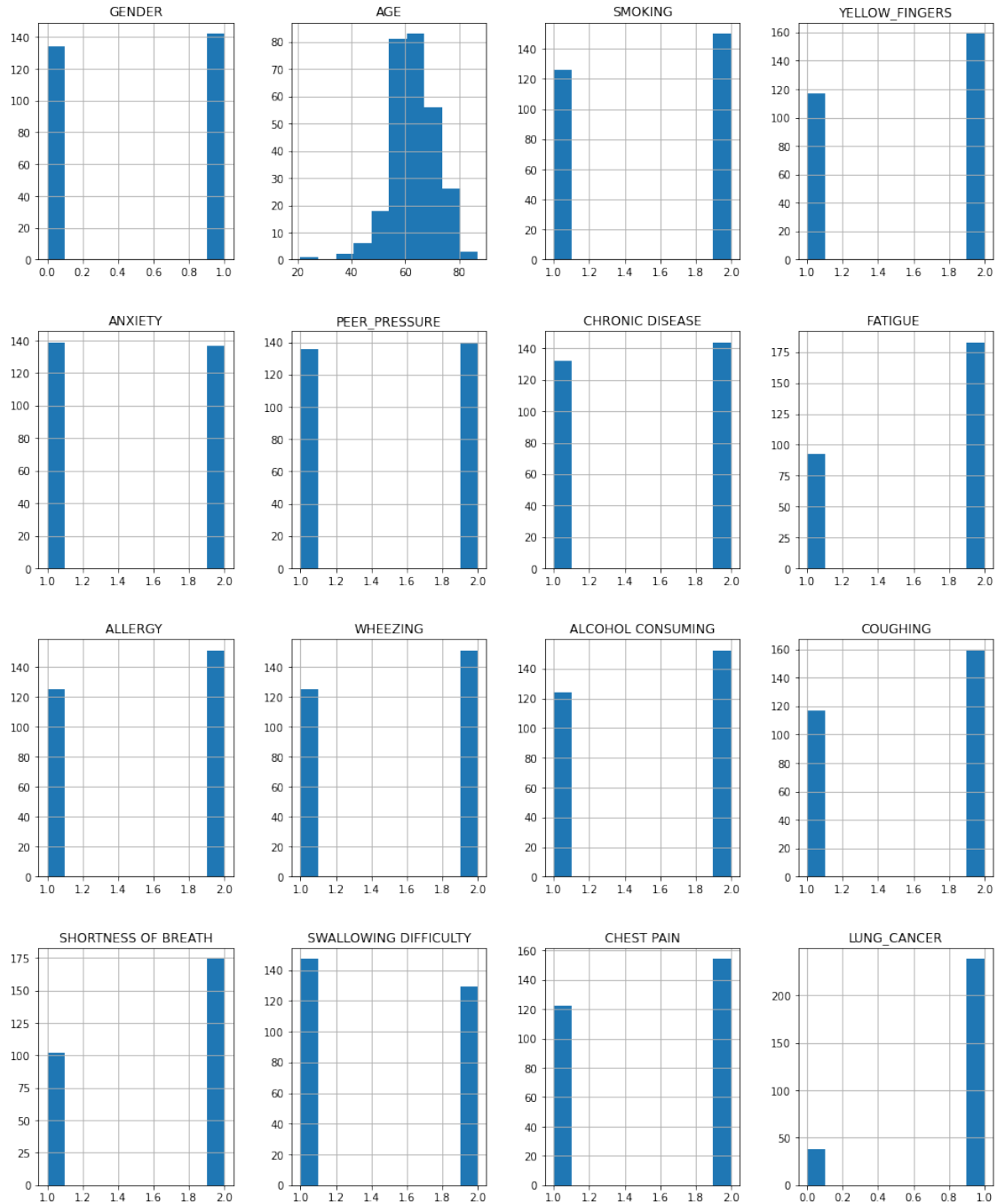
```
Number of duplicates:  33
Number of duplcates after drop  0
```

1.1 Histrogram

```
[ ]: # Data histogram
     df.hist(figsize=(16,20))
```

```
[ ]: array([[<AxesSubplot:title={'center':'GENDER'}>,
             <AxesSubplot:title={'center':'AGE'}>,
             <AxesSubplot:title={'center':'SMOKING'}>,
             <AxesSubplot:title={'center':'YELLOW_FINGERS'}>],
            [<AxesSubplot:title={'center':'ANXIETY'}>,
             <AxesSubplot:title={'center':'PEER_PRESSURE'}>,
             <AxesSubplot:title={'center':'CHRONIC DISEASE'}>,
             <AxesSubplot:title={'center':'FATIGUE '}>],
            [<AxesSubplot:title={'center':'ALLERGY '}>,
             <AxesSubplot:title={'center':'WHEEZING'}>,
             <AxesSubplot:title={'center':'ALCOHOL CONSUMING'}>,
             <AxesSubplot:title={'center':'COUGHING'}>],
            [<AxesSubplot:title={'center':'SHORTNESS OF BREATH'}>,
             <AxesSubplot:title={'center':'SWALLOWING DIFFICULTY'}>,
             <AxesSubplot:title={'center':'CHEST PAIN'}>,
             <AxesSubplot:title={'center':'LUNG_CANCER'}>]], dtype=object)
```

1.2 Heatmap

```python
# create X and y sets
X = df.drop(columns="LUNG_CANCER").to_numpy()
X_ht = df.drop(columns="LUNG_CANCER", axis= 1)
y = df['LUNG_CANCER'].to_numpy().reshape(-1,)
```

```
# Heatmap of dataframe
plt.figure(figsize=(16, 6))
mask = np.triu(np.ones_like(X_ht.corr()))
htmap = sns.heatmap(X_ht.corr(),vmin= -1, vmax= 1,  annot= True , mask = mask)
htmap.set_title('Triangle Correlation Heatmap', fontdict={'fontsize':18},
    →pad=16);

#https://medium.com/@szabo.bibor/
    →how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e
```


Triangle Correlation Heatmap
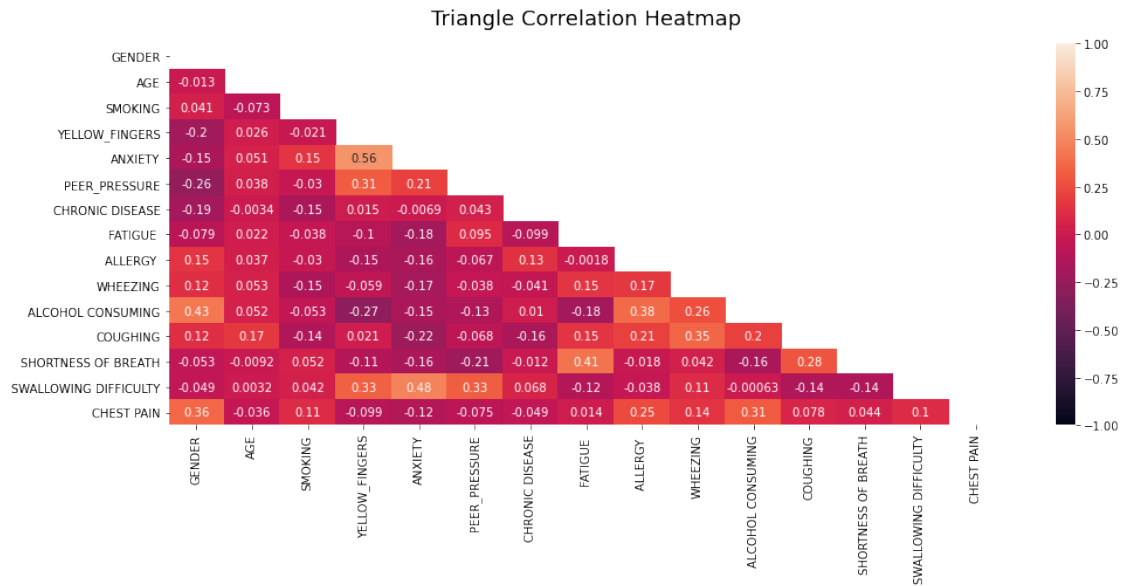
1.3 Virtual Inflation Factor VIF

Was just used for some manual testing

```
X_t = df.drop(columns="LUNG_CANCER", axis=1)
vif_data = pd.DataFrame()
vif_data["feature"] = X_t.columns
vif_data["VIF"] = [variance_inflation_factor(X_t.values, i) for i in
    →range(len(X_t.columns))]
print(vif_data.sort_values(by = ["VIF"]))

#https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/
```

```
         feature        VIF
0         GENDER    3.012213
2        SMOKING   10.738372
6  CHRONIC DISEASE  10.896006
8        ALLERGY   13.490502
```

```
5              PEER_PRESSURE   13.564503
9                   WHEEZING   13.986291
14                CHEST PAIN   14.003207
13      SWALLOWING DIFFICULTY   15.218347
11                  COUGHING   17.344194
10         ALCOHOL CONSUMING   17.545363
12       SHORTNESS OF BREATH   17.666498
7                    FATIGUE   17.770848
3             YELLOW_FINGERS   19.138132
4                    ANXIETY   19.508213
1                        AGE   41.961929
```

2 Machine Learning part

```python
# Spitting the data in 5 sets
kf = KFold(n_splits=5, shuffle=True, random_state=69)

#results for train error
train_res_acc = []
train_res_err = []

# results for validation set
val_res_acc = []
val_res_err = []

# iteration for k-fold
for train_index, val_index in kf.split(X):
    X_train, X_val= X[train_index], X[val_index]
    y_train, y_val= y[train_index], y[val_index]

    # make logistic regression model
    linreg = LogisticRegression(max_iter=10000)
    linreg.fit(X_train, y_train)

    # predict training set
    y_pred_train = linreg.predict(X_train)

    # calculate accuracy and logistic loss
    train_acc = accuracy_score(y_train, y_pred_train)
    train_err = log_loss(y_train, y_pred_train)

    # append results for later
    train_res_acc.append(train_acc)
    train_res_err.append(train_err)

    # predict validation set
    y_pred_val = linreg.predict(X_val)
```

```python
    # calculate accuracy and logistic loss
    val_acc = accuracy_score(y_val, y_pred_val)
    val_error = log_loss(y_val, y_pred_val)

    # append results for later
    val_res_acc.append(val_acc)
    val_res_err.append(val_error)


# print results
print("Training accuracy: ", np.mean(train_res_acc))
print("Training error: ", np.mean(train_res_err))
print("Validation accuracy: ", np.mean(val_res_acc))
print("Validation error: ", np.mean(val_res_err))
```

```
Training accuracy:  0.9248087206910738
Training error:  2.597054627913788
Validation accuracy:  0.9092857142857141
Validation error:  3.1332097559252547
```

[ ]: