Predicting Lung Cancer with Machine Learning

September 23, 2022

1 Problem statement

In this paper I will try to predict whether a patient with certain characteristics has lung cancer or not.

1.1 Data points

The data set consists of 309 entries in an Excel spreadsheet, with patients between 21 and 87 years old. Each data point describes a patient with different characteristics. The dataset is licensed under the *CCO: Public Domain* and was found on *Kaggle* [1]. For transparency, this dataset is used in another project found on *here* [2], but as this work was done independently, it has not been copied in any way. The dataset is complete and has no missing features. Each data point is mapped with 16 different attributes (gender, age, smoking, yellow fingers, chronic disease, *see appendics for more...*).

The label for this prediction is whether a patient has lung cancer or not (yes/no), which means that this problem is a classification problem. I decided to change the dataset for the labels from *yes* and *no* to *-1* and *1* to make the prediction easier and because I will use the SVC hinge loss function in the second part. All attributes except gender and age are represented with *1* or *2*, which corresponds to *no* and *yes* respectively. Gender is represented with *M* or *F*, which means male or female respectively. The age is the actual age in the form of a number. The histogram shows that the data is largely balanced, except for a few outliers such as *fatigue* and *lung cancer*. Since I will be using logistic regression, these outliers will not significantly affect our results [3][4].

Additionally, I was also able to find 33 duplicates, which I have eliminated. As you may have already noticed, the dataset is not too large, now containing only 276 entries.

1.2 Feature selection

For the characteristics, I selected all attributes except for the lung cancer column, as this is used for the label. For the selected characteristics, I have changed the data points from *1* and *2* to *0* and *1* for convention and ease of understanding. It is important to note here that with the exception of age, all characteristics can be represented in binary, meaning, a normalisation is not required. After looking at the correlation heat map of the characteristic, I could not identify any bad data sets. All data points appear to be in the range of -0.75 and 0.75, which is a common indicator of correlation. Also, all the identifiers, gender, age, smoking, yellow fingers, (...) seem intuitively important in determining whether a patient has lung cancer or not. After some small tests with *Variance*

Inflation Factor (VIF) I could confirm that all 15 characteristics seem to improve the accuracy of the prediction.

1.3 Data breakdown

As mentioned earlier, the data set is not very large, so I decided to use the k-fold cross-validation method. This method is suitable for data sets with fewer entries. The algorithm works by randomly dividing the datasets into equal parts and using one set as the test set and the rest as the training set. I decided to split the data into 5 parts (k=5) to achieve an 80% to 20% ratio, which is very common.

1.4 Machine learning model

Our goal is to classify whether a patient has lung cancer. This can be described as a binary classification problem.

For the first machine learning model, I chose logistic regression because it seemed reasonable for a simple binary classification problem. Logistic regression uses a linear hypothesis space and works by setting a limiter between the data points. In a 2D space, you would put a line between the given data points. In 3D space, the points would be separated by a plane, and in higher dimensions you would use a hyperplane to describe the separation. For the loss function, I chose the logistic loss function, as it is a very common and proven function for logistic regression and was easy to use as it is already implemented in the used library.

2 References

- [1] Data set from Kaggle: `https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer`
- [2] Other Kaggle project with same data set: `https://www.kaggle.com/code/gaganmaahi224/lung-cancer-5ml-models-full-analysis-plotly`
- [3] How to handle unbalanced sets tutorial : `https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html`
- [4] Unbalanced data in Logistic Regression: `https://stats.stackexchange.com/questions/6067/does-an-unbalanced-sample-matter-when-doing-logistic-regression`
- [5] Heatmap Tutorial *Medium* an seaborn library: `https://medium.com/@szabo.bibor/how-to-\\create-a-seaborn-correlation-heatmap-in-python-834c0686b88e`

3 Code Appendics

Predicting Lung Cancer with Machine Learning

September 23, 2022

[ ]:
```
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
```

[ ]:
```
# 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
```

[ ]:
```
# 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

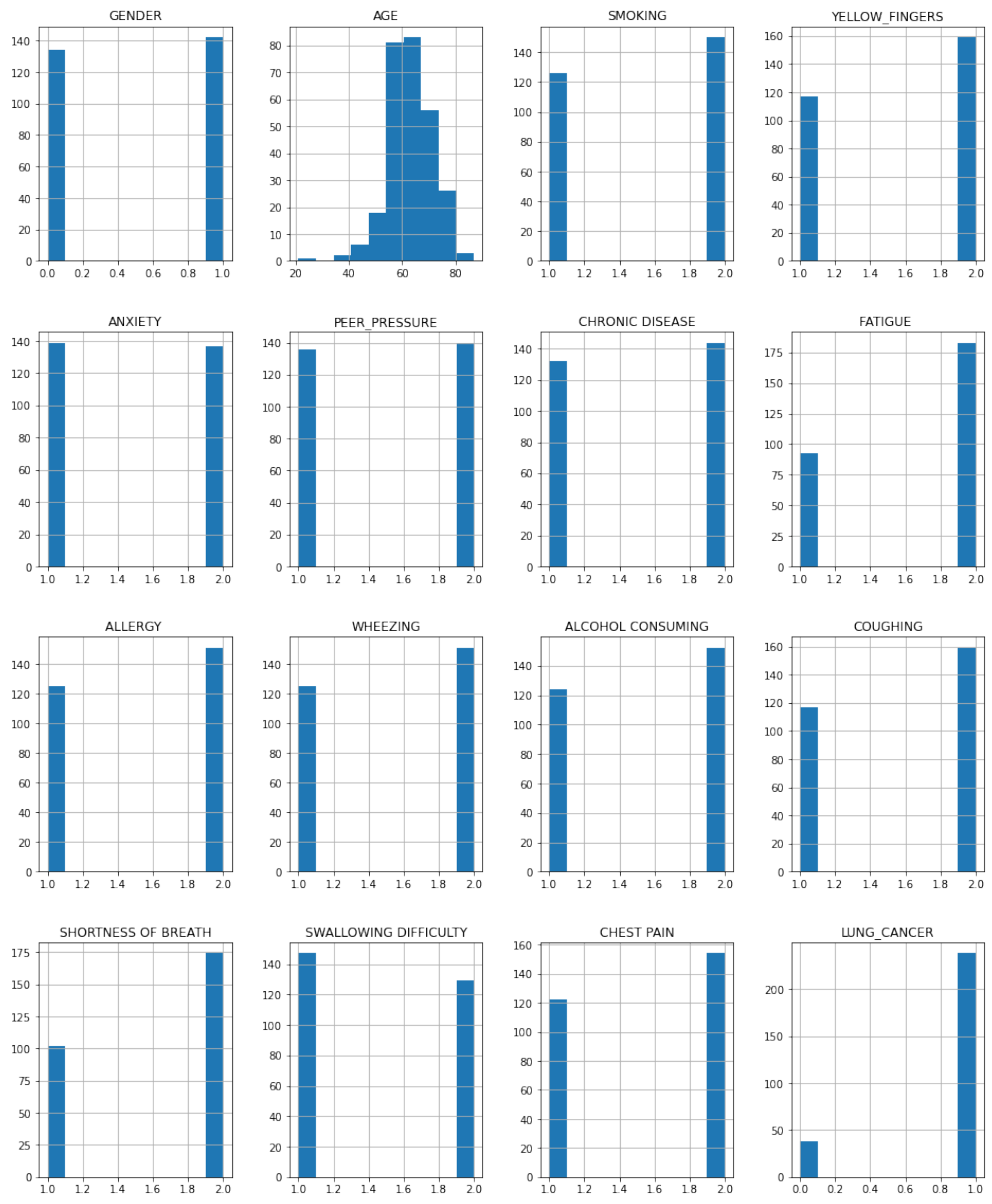
```
[ ]: print("Number of duplicates: ", df.duplicated().sum())
     df = df.drop_duplicates()
     print("Number of duplcates after drop ", df.duplicated().sum())
```

```
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)
```

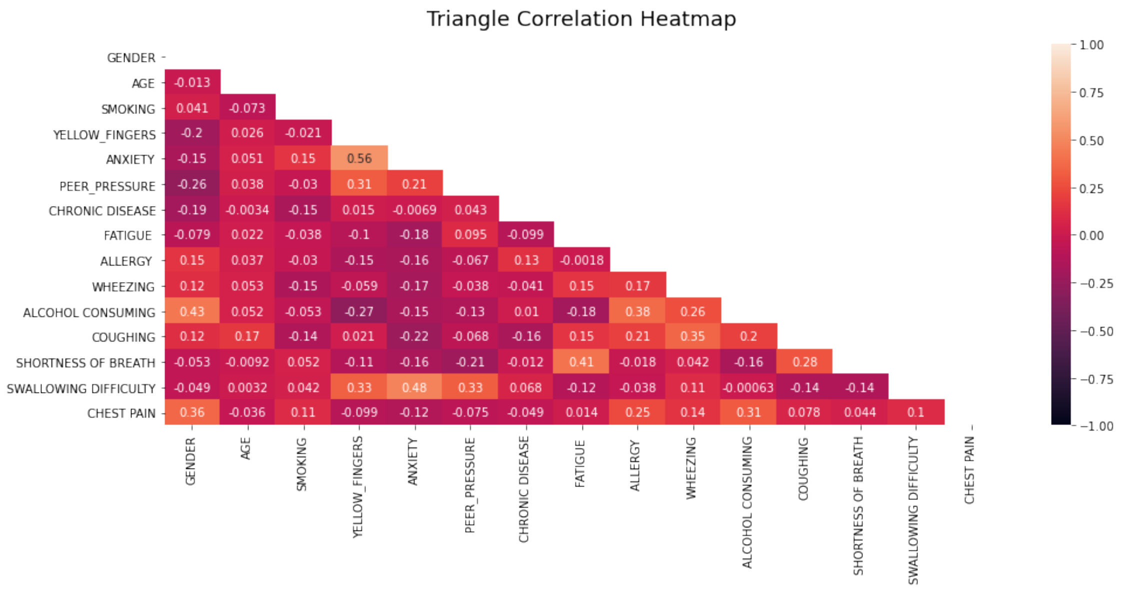


1.2 Heatmap

```
[ ]: # 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
```



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)
    # 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
```

```
[ ]:
```