Predicting Lung Cancer with Machine Learning

October 4, 2022

1 Introduction

Lung cancer is the most common form of cancer in the whole world. Over 200.000 new cases are being diagnosed every year which makes up to 13% of of all cancer diagnosis. Almost half of the patients with lungcancer die after a year after the diagnosis. As for all forms of cancer, an early diagnosis increases the survival rate drasticaly. Earlier stages of lung cancer are easier to treat and can decrease the death rate by 14% to up to 20%[1]. Applying machine learning appears to be reasonable use case for diagnosis of lung cancer, since it can achieve high effectivnes. In this paper I will try to determine whether a patient with certain characteristics has lung cancer or not.

1.1 Report structure

The report will continue with the section 2 *"Problem Statement"*, where the data set presented and all the data points are clarified. Section 3 *"Methods"* will explain how and why the feature selction was done and how the data sets were constructed. Furthermore, the two machine learning methods, logistic regression and SVC, will be discussed. In section 4 *"Results"* the results are evaluated and section 5 *"Conclusion"* will draw a final conclusion.

2 Problem Formulation

2.1 Data set

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* [2]. For transparency, this dataset is used in another project found on *here* [3], 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...*). 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. How we dealt with this problem will be further explained in the "Data Splitting" section

2.2 Label and feature clarification

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 [4][5].

3 Methods

3.1 Feature selection and engineering

For the features, I selected all attributes except for the lung cancer column, as this is used for the label. For the selected features, 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.

3.2 Data Splitting

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.

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

3.3 Logistic Regression

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.

3.3.1 Loss function for Logistic Regression

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.

3.4 SVC

For the second machine learning model, we chose to use the Support Vector Machine (SVM) classifier class, Support Vector Classification (SVC), for this simple binary classification task.

For SVC, it also uses a linear hypothesis space that maps $h(x) = w^T(x)$, identical to that of the logistic regression method used previously. The decision to opt for this method was so that we can compare the classification methods and evaluate which has better performance. We did not choose to make use of LinearSVC or SGDClassifier over the basic SVC due to the dataset being not too large.

3.4.1 Loss function for SVC

We also decided on using hinge loss to calculate the loss of our SVC method. The motivation for making use of hinge loss as the function for this method % hinge loss function ,% is mainly because it is widely known as the loss function tailored for SVM and also is easily accessible with sklearn.metrics. Furthermore, it also makes sense to use this loss function for this binary classification task. A visualization and representation of the hinge loss function is shown in the following figures.

4 Results

From these figures, we can see that for observations that are of a margin distance of greater than or equal to 1, the hinge loss is valued at zero. While for observations of margin distance less than 1, the hinge loss value incurs a loss the increases linearly. To put simply, while the SVC bears the similarity with logistic regression in that it aims to separate both classes with a line, the difference lies in this hinge loss function, that aims to maximize the margin distance between each data point and the separating line. The dataset for the SVC method is also split into 5 parts to achieve the same 80% training to 20% validation ratio, for a fairer comparison between the two methods. We also decided to use a linear kernel for the SVC given that the nature of our data.

To evaluate and compare the two models, we have calculated the errors and the accuracy scores for each training and validation sets, obtaining the following results shown in the charts below. As we can see from the charts and table above, both methods performed well in this binary classification to predict persons with lung cancer, with the training and validation accuracies for both Logistic Regression and SVC valued above 90%. It can also be observed that for both training and validation, SVC performed better than the Logistic Regression model, with a 93.5% training and 91.7% validation accuracy, compared to the 92.5% training and 90.9% validation accuracy of the latter.

Additionally, from our results, we can clearly see how the absolute errors for SVC using hinge loss is significantly smaller than that of Logistic Regression. From these results, we therefore come to the decision that SVC is the better method for this prediction of lung cancer binary classification task.

5 Conclusion

Conclusion: briefly summarise the report and interpret the results; discuss if the obtained results seem to be optimal or if there is room for improvement speculate about future directions on how to further

6 References

- [1] Lung cancer fact sheet website: `https://www.lung.org/lung-health-diseases/lung-disease-lookup/lung-cancer/resource-library/lung-cancer-fact-sheet`
- [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`

7 Code Appendics

jupyternotbook_code

October 4, 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, hinge_loss
from sklearn.model_selection import train_test_split, KFold
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.svm import SVC
```

[ ]:
```
# 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

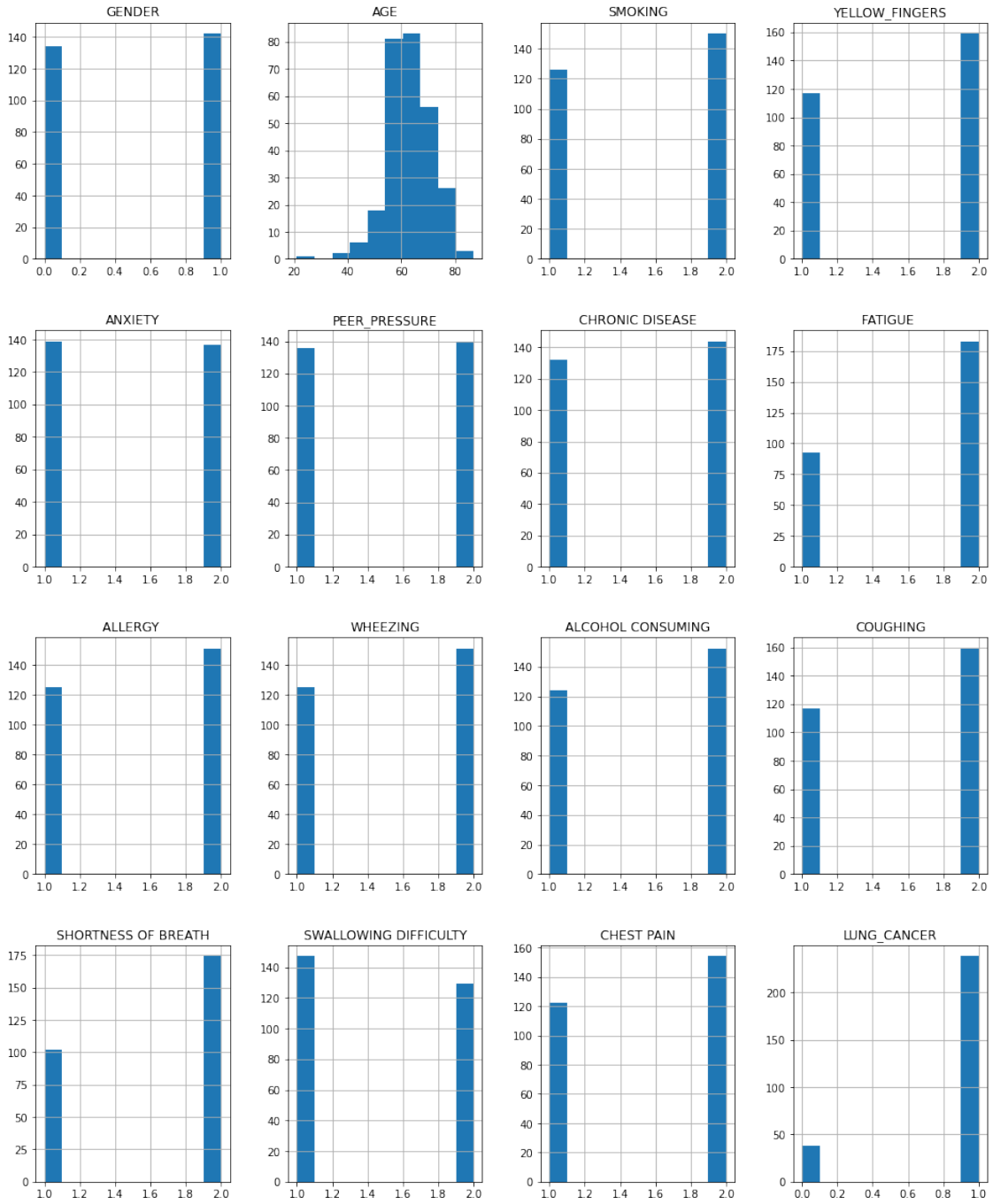
[ ]:
```
print("Number of duplicates: ", df.duplicated().sum())
df = df.drop_duplicates()
print("Number of duplcates after drop ", df.duplicated().sum())
```

```
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)
```

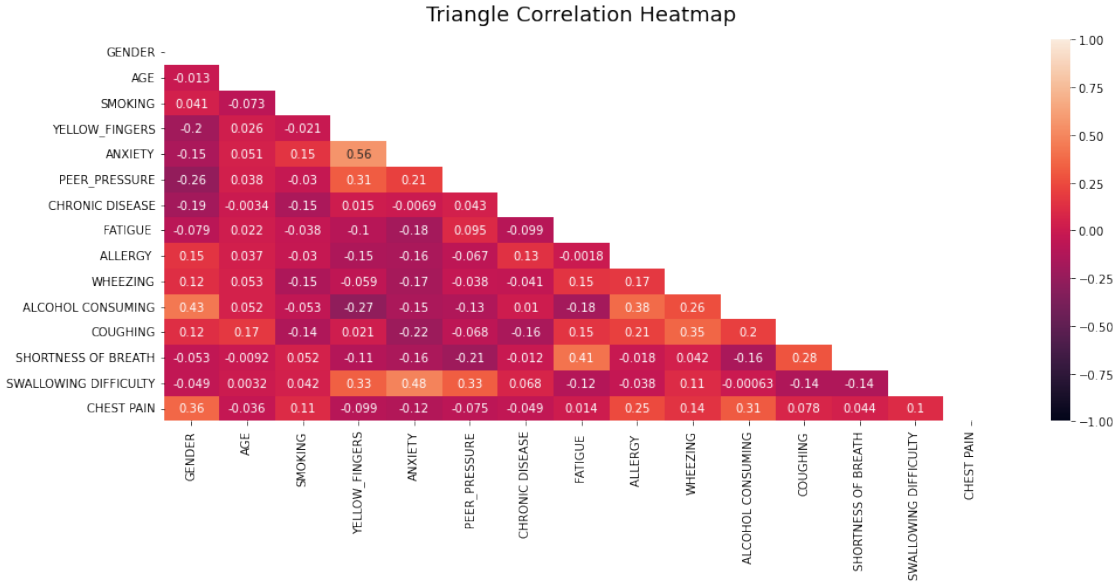


1.2 Heatmap

[ ]:
```
# 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

2.1 Logistic Regression

[ ]:
```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=100000)
    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
```

2.2 SVC

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

#results for train error
train_res_acc2 = []
train_res_err2 = []

# results for validation set
val_res_acc2 = []
val_res_err2 = []

# 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 SVC model
    svc = SVC(gamma='auto',kernel='linear')
    svc.fit(X_train,y_train)

    # predict training set
    y_pred_train2 = svc.predict(X_train)
```

```
    # calculate accuracy and logistic loss
    train_acc2 = accuracy_score(y_train, y_pred_train2)
    train_err2 = hinge_loss(y_train, y_pred_train2)

    # append results for later
    train_res_acc2.append(train_acc2)
    train_res_err2.append(train_err2)

    # predict validation set
    y_pred_val2 = svc.predict(X_val)

    # calculate accuracy and logistic loss
    val_acc2 = accuracy_score(y_val, y_pred_val2)
    val_error2 = hinge_loss(y_val, y_pred_val2)

    # append results for later
    val_res_acc2.append(val_acc2)
    val_res_err2.append(val_error2)


# print results
print("Training accuracy: ", np.mean(train_res_acc2))
print("Training error: ", np.mean(train_res_err2))
print("Validation accuracy: ", np.mean(val_res_acc2))
print("Validation error: ", np.mean(val_res_err2))
```

```
Training accuracy:  0.9356849033319623
Training error:  0.20199506375976967
Validation accuracy:  0.9166233766233767
Validation error:  0.22103896103896106
```

[ ]: