1.04-CK-Analyse

November 15, 2024

[1]:
```
import numpy as np
import pandas as pd
import scipy
from matplotlib import pyplot as plt
import os
import sys
sys.path.append(os.path.dirname(os.getcwd()))
from src.load_covid19 import load_clean_covid19
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
from imblearn.over_sampling import SMOTE
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

[2]:
```
current_dir = os.getcwd()
file_path = os.path.join(os.path.dirname(current_dir), "data", "raw",␣
 ↪"covid19-dataset", "Covid Data.csv")
df = pd.read_csv(file_path)
```

[3]:
```
def get_scores(y_test, y_pred):
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    return acc, f1, prec, rec
```

[4]:
```
#Wir fügen zuerst ein neues Attribut hinzu, das angibt, ob die Person gestorben␣
 ↪ist oder nicht.
#Die Verschlüsselung erfolgt konsistent mit den anderen Attributen (2 für␣
 ↪"nein", 1 für "ja").
#Das Todesdatum wird als Attribut verworfen.

df['DIED'] = [2 if i=='9999-99-99' else 1 for i in df.DATE_DIED]
```

```
df=df.drop(columns='DATE_DIED')
```

[5]:
```
#Eine Analyse der Missing Values zeigt, dass diese vor allem in den Spalten␣
 ↪"PREGNANT", "ICU" und "INTUBED" vorkommen.
#Das Attribut "PREGNANT" wird bei allen männlichen Patienten auf 2 gesetzt.
#Das Attribut "PATIENT_TYPE" gibt an, ob die Person hospitalisiert war oder␣
 ↪nicht. Bei allen Patienten ohne
#Krankenhausaufenthalt kann das Attribut "ICU" auf 2 gesetzt werden, da keiner␣
 ↪dieser Patienten folglich auf
#einer Intensivstation behandelt wurde.
#Dieselbe Vorgehensweise wenden wir auf das Attribut "INTUBED" an. Dazu gehen␣
 ↪wir von der Grundannahme aus, dass
#ein Anschluss an ein Beatmungsgerät im Rahmen eines Klinikaufenthaltes erfolgt.
 ↪ Etwaige Fälle von Heimintubation
#vernachlässigen wir dabei.

df.loc[df.SEX==2,'PREGNANT']=2
df.loc[df.PATIENT_TYPE==1,'ICU']=2
df.loc[df.PATIENT_TYPE==1,'INTUBED']=2
```

[6]:
```
#Für die anderen fehlenden Werte können keine sinnvollen Aussagen getroffen␣
 ↪werden. Da sie nur einen sehr kleinen
#Anteil des gesamten Datensatzes ausmachen (2.76%), verwerfen wir sie.

for col in df.columns.drop('AGE'):
    for i in [97,98, 99]:
        df[col]=df[col].replace(i , np.nan)

df=df.dropna()
df=df.astype('float64')
```

[7]:
```
#Als Zielvariable wählen wir uns 'DIED' aus.
#Wir testen die übrigen Attribute auf Korrelation mit der Zielvariablen und␣
 ↪verwerfen
#alle Attribute, die unter einem gewissen Schwellwert liegen. (->andere Feature␣
 ↪Selection Methoden ausprobieren)

target='DIED'
threshold=0.04
selected_features=df.corr()[target][abs(df.corr()[target])>threshold].index
df=df[selected_features]
```

[8]:
```
#Wir teilen den Datensatz in Trainings- und Testdatensatz auf.

test_size=0.2
train, test = train_test_split(df, test_size=test_size, shuffle=True)
```

```
train_y = train[target]
train_x = train.drop(target,axis=1)
```

[9]:
```
#Da die Werte der Zielvariablen unbalanciert sind, wenden wir ein␣
 ↪SMOTE-Oversampling an,
#um ein balanciertes Trainingsset zu erreichen. (-> auch Undersampling␣
 ↪probieren)

sm = SMOTE()
train_x, train_y = sm.fit_resample(train_x, train_y)

test_y = test[target]
test_x = test.drop(target,axis=1)
```

[10]:
```
#Wir wenden drei "herkömmliche" Methoden zum maschinellen Lernen eines␣
 ↪Klassifikators an:
#Naive Bayes, Entscheidungsbäume bzw. Random Forests und die Logistische␣
 ↪Regression.
#(->verschiedene Parameter für die Klassifier ausprobieren und optimieren)
#(->SVM und KNN eventuell auch ausprobieren)

for clf in␣
 ↪[GaussianNB(),DecisionTreeClassifier(),LogisticRegression(max_iter=500)]:
    #if clf==LogisticRegression():
        #train_x = StandardScaler().fit(train_x)
        #test_x  = StandardScaler().fit(test_x)
    clf.fit(train_x,train_y)
    y_pred=clf.predict(test_x)
    acc, f1, prec, rec = get_scores(test_y, y_pred)
    print(f"Klassifikator: "+type(clf).__name__+"\n")
    print(f'Accuracy: {acc}')
    print(f'F1: {f1}')
    print(f'Precision: {prec}')
    print(f'Recall: {rec}'+"\n")
```

```
Klassifikator: GaussianNB

Accuracy: 0.8898516186609393
F1: 0.5266861922922944
Precision: 0.38215508806262233
Recall: 0.8470349034225686

Klassifikator: DecisionTreeClassifier

Accuracy: 0.9279619877019035
F1: 0.5699481865284974
Precision: 0.5016489745439555
```

```
Recall: 0.6597763470010166

Klassifikator: LogisticRegression

Accuracy: 0.8979964106034305
F1: 0.5640456031519826
Precision: 0.4082703801462334
Recall: 0.9120298203998645
```

[ ]: