Using tree-based gradient boosting to distinguish between lymphoma and COVID-19

MD PHOLO

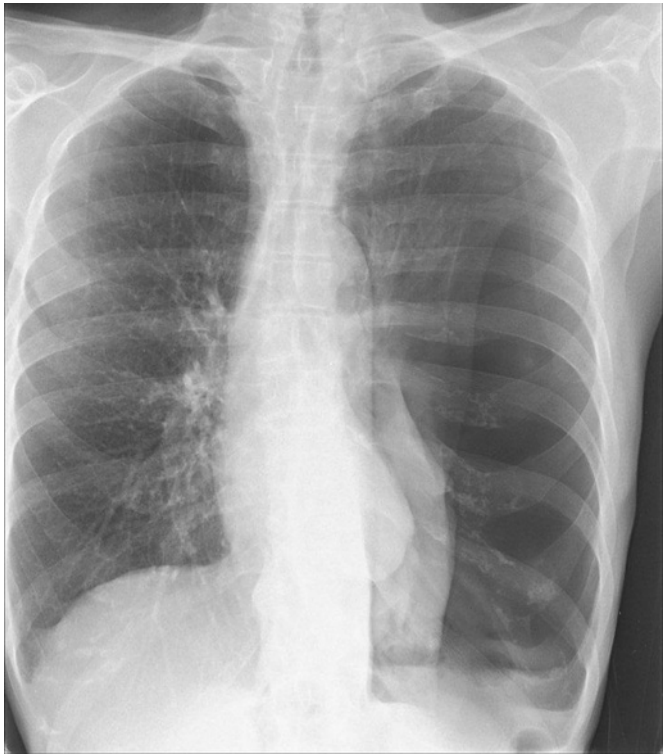
(Tshwane University of Technology)

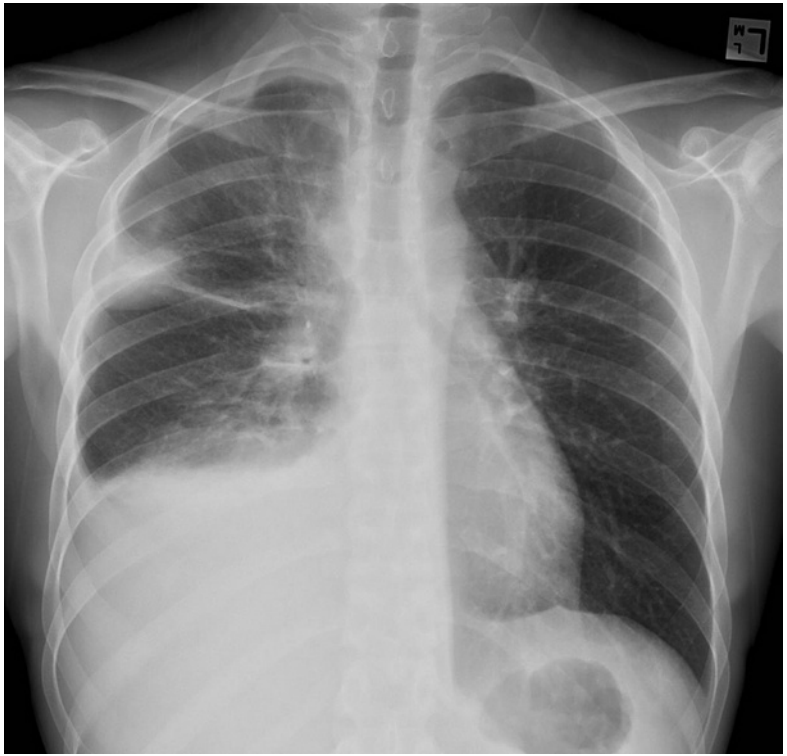
INTRODUCTION

Problem

- Lymphomas = group of cancers that occur when lymphocytes multiply at unusual rate or live too long
- Usual symptoms: enlarged lymph nodes, fever, fatigue, shortness of breath, night sweats

Problem (cont.)





Problem (cont.)

- Challenge: late diagnosis → lower prognosis
- Recent COVID-19 pandemic caused drastic changes in healthcare systems
  - → Potentially higher delay in lymphoma diagnosis

Proposed Solution

- Model that differentiates between COVID-19 and lymphoma
  - Features used: age, gender, clinical symptoms
- Potential: could serve as screening tool at COVID-19 stations to reduce delayed lymphoma diagnosis.

METHODOLOGY

Data Collection

- Dataset = lymphoma and COVID-19 case reports collected in previous studies
- Case report = description of clinical events and conditions leading to diagnosis of a single patient
- Examining multiple cases together can increase findings' validity

Data Collection (cont.)

- Tool 1: import **urllib.request**
- ***urllib*** is a package that collects several modules for working with URLs
- The ***urllib.request*** module helps make URL requests

```
numArticle=1
testbody = None
for count in range(0,5000,25):
    url = 'https://api.elsevier.com/content/search/sciencedirect?'
    apikey = '44                                    8c'
    query = 'lymphoma+case+report'
    url = url + 'query='+ query +'&apiKey=' + apikey + '&start=' + str(count)
```

Data Collection (cont.)

- Tool 2: import **json**
- Sciencedirect returns JSON results
- *json.loads ()* method parses valid JSON string into Python Dictionary.

```
with urllib.request.urlopen(url) as response:
    result = json.loads(response.read().decode("utf-8"))
```

Data Collection (cont.)

- Tool 3: import **BeautifulSoup**

```
with urllib.request.urlopen(url) as response:
    result = json.loads(response.read().decode("utf-8"))
    for i in range(len(result['search-results']['entry'])):
        pii=result['search-results']['entry'][i]['pii']
        title=result['search-results']['entry'][i]["dc:title"]

        if (title.lower().find('case')!=-1
            or title.lower().find('manifestation')!=-1 ):
            #open article URL
            u="https://api.elsevier.com/content/article/pii/"+pii+"?apiKey=444
            h = urllib.request.urlopen(u).read()
            s = BeautifulSoup(h, 'lxml')
            body=s.find("body")
```

Data Collection (cont.)

- Example data point

2. Case

This case describes a 17-year-old female with Cornelia de Lange syndrome and well controlled epilepsy with infrequent brief tonic-clonic seizures; who at neurological baseline was able to mobilise by shuffling her body from a seated position and to indicate her needs non-verbally.

She presented with cough, fever and difficulty in breathing. Due to an increasing oxygen requirement she was initially started on high flow oxygen but continued to require FiO2 80 % to maintain SaO2 above 94 %. In view of her work of breathing and oxygen requirement a decision was made to intubate and transfer to the paediatric critical care unit. She showed signs of sepsis (spiking fevers, HR 120–130bpm, BP systolic 120 mmHg, capillary refill time <2 s). Her bloods showed CRP 275 mg/L, ferritin 2091 microgram/liter, deranged clotting (PT 13.4 s, INR 1.3, APTT 38 s, fibrinogen 3.75 g/L) and bone marrow failure (Hb 73 g/L, plt 51 × 10^9/L, WCC 2.7 × 10^9/L). A chest x-ray showed bilateral hilar consolidation.

Pre-processing

- Tool 5: Amazon Medical Comprehend
  - Extracted from every case report using **Amazon Medical Comprehend** API
    - web service used to extract entities such as diseases, medicines and symptoms from medical text
    - Uses NLP to detect entities such as medical conditions, medications

Pre-processing

```
[['cough',
  'fever',
  'difficulty in breathing',
  'hilar consolidation',
  'Respiratory secretions'],
 ['fever', 'fatigue', 'mechanically ventilated'],
 ['dyspnea',
  'fever',
  'fully conscious',
  'exophtamia',
  'extraocular movements limited',
  'dilated left pupil'],
 ['dyspnea',
```

Pre-processing (cont.)

- Tool 6: **nltk.stem.PorterSterm**
  - nltk = Natural Language Toolkit = library for working with human language data.
  - Stemming = process of producing morphological variants of a root/base word.
  - But why?

Pre-processing (cont.)

```
# Stem
from nltk.stem import PorterStemmer

def stem(tokens):
    ps = PorterStemmer()
    new_tokens = []
    for token in tokens:
        words = token.split()
        words = [ps.stem(w) for w in words]
        token = " ". join(words)
        new_tokens.append(token)
    return new_tokens
```

Pre-processing (cont.)

4kg in 2 week	abdomen wa soft	abdomin distens	abdomin pain	abdomin tender	abl to walk	abnorm jerki movement	abras	absent deep tendon reflex
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Pre-processing (cont.)

- Dealing with “duplicate” symptoms:
  - Regex
  - Manual list of synonyms

Pre-processing (co

- Tool 7: RegEx
- A RegEx = Regular Expression = sequence of characters that forms search pattern.
- Can be used to check if string contains specified pattern.

Pre-processing (cont.)

- Age extracted using RegEx

```
import re
def search_pattern (txt):
    pattern = re.findall(r'\d+.years?[ \-]old', txt)
    if pattern:
        age = re.findall(r'\d+', pattern[0])
    return age[0] if pattern else '-1'
```

Pre-processing (cont.)

- Gender extracted using RegEx

```
def extract_gender (word):
    if (word.find(' male') != -1) or word.find(' man') != -1
    or word.find('boy') != -1 or word.find('gentleman') != -1:
        return '0'
    if (word.find('female') != -1)
    or word.find('girl') != -1 or word.find('woman') != -1 or word.find('lady')
        return '1'
    else:
        return '-1'
```

Feature Selection

- Based on correlation
- Correlation expresses strength of relationship between two variables
  - Correlation coefficient quite close to 0

```
data = df_symp.drop(['Unnamed: 0', 'PII', 'age_below_average', 'Label'], axis = 1)
corr = data.corr()
corr = pd.melt(corr.reset_index(), id_vars='index') # Unpivot the dataframe,
                                                    # so we can get pair of arrays for x and y
corr.columns = ['x', 'y', 'value']
```

```
to_remove = corr[(corr.value >= 0.7) | (corr.value <= -0.7)][corr.x != corr.y].x.unique()
```

Modelling using tree-based methods

- **Decision trees** = popular classification and regression algorithm
- Advantages:
  - easy to interpret
  - filter out unimportant features
- Main disadvantage: Prone to overfitting
- One solution: combine decision trees into ensemble classifiers to obtain better performance

Modelling using tree-based methods

- Boosting = ensemble method which combines predictions of multiple weak classifiers
  - makes decision based on majority vote
  - Classifiers are built stage-wise
- **XGBoost** (XGB) is a gradient boosting algorithm
  - reduces overfitting using shrinkage and column subsampling

Modelling using tree-based methods

- **Catboost** : inherently deals with categorical variables
- Catboost limits number of categories to be converted into binary features
- If a feature has more categories than threshold, CatBoost will sample dataset and performcreate binary variables using smaller sample

Modelling using tree-based methods

- **LightGBM** = efficient tree-based boosting algorithm
- Aims to improve processing and memory resource usage
- Grows decision trees leaf-wise rather than depth-wise.
  - splitting focuses on first splitting node that most reduces loss
  - Instead of nodes' proximity to root node

Modelling using tree-based methods

- **LightGBM** = efficient tree-based boosting algorithm
- Aims to improve processing and memory resource usage
- Grows decision trees leaf-wise rather than depth-wise.
  - splitting focuses on first splitting node that most reduces loss
  - Instead of nodes' proximity to root node

Modelling using tree-based methods

```
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
```

Model Evaluation

- Each algorithm was tuned by training model using different hyper-parameter values
- An ROC curve (receiver operating characteristic curve) = graph showing performance of classification model

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = 1 - \frac{TN}{FP + TN}$$

Where:
TP = true positive (correct hit)
TN = true negative (correct rejection)
FP = false positive (incorrect hit)
FN = false negative (incorrect rejection)

Model Evaluation

```
params_classifiers = {
    "Decision Tree": {
        'max_depth': (5, 10, 18, 20, 25, 30, 50 ),
        'max_features': (10, 50, 75, 100, 125, 148)
    },
    "Catboost" : {
        'subsample': [0.8, 0.9, 1.0],
        'depth'          : [4,5,6,7,8,9, 10],
                'learning_rate' : [0.01,0.02,0.03,0.04],
                 'iterations'    : [10, 20,30,40,50,60,70,80,90, 100],
        'l2_leaf_reg': [1, 3, 5,],
                },
    "XGBoost": {
        'eta':[0.05,0.1],
        'subsample': [0.8, 0.9, 1.0],
        'colsample_bytree': [0.4, 0.5, 0.6],
        'n_estimators': [1, 5, 10, 20, 50],
         'max_depth': [-1, 5, 10, 20],
    },
    "LGBM":
    {'num_leaves':[20,50,75,100],
     'min_child_samples':[5,10,15],
     'max_depth':[-1,5,10,20],
     'learning_rate':[0.05,0.1],
     'reg_alpha':[0,0.01]}
}
```

Model Evaluation

- GridSearchCV: Exhaustive search over specified parameter values for an estimator.

```
from sklearn.model_selection import GridSearchCV
```

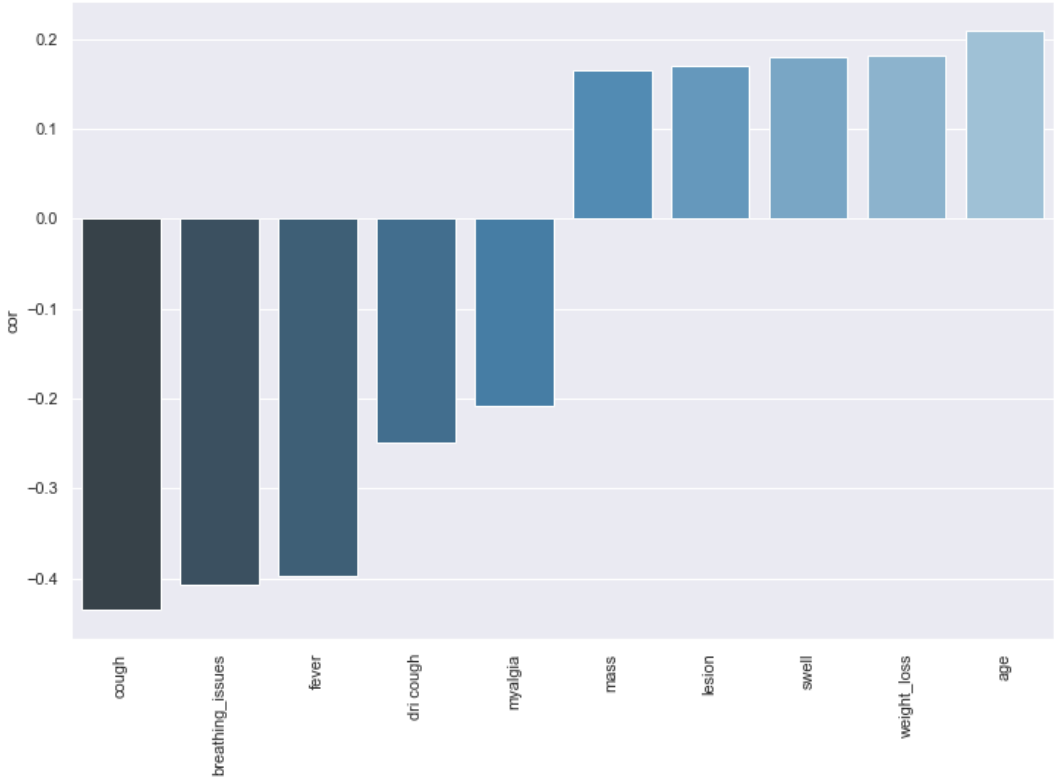
```
for name, classifier in zip(names, classifiers):

        print(name)

        param_search = None

        if score != 'accuracy':
            param_search = GridSearchCV(
            estimator=classifier, param_grid=params_classifiers[name], scoring = score)
```

RESULTS

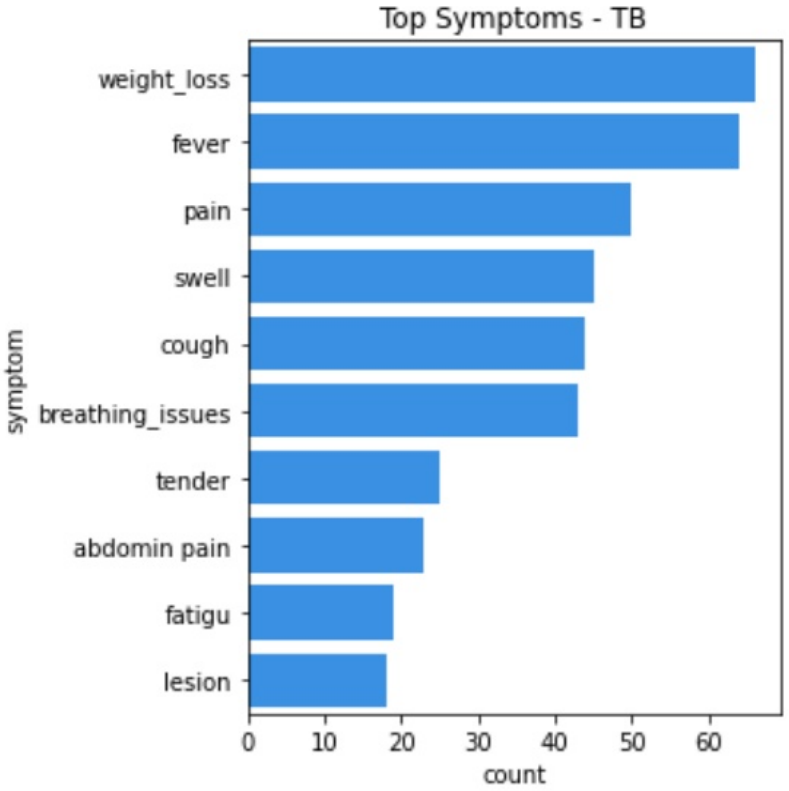
Exploratory Analysis

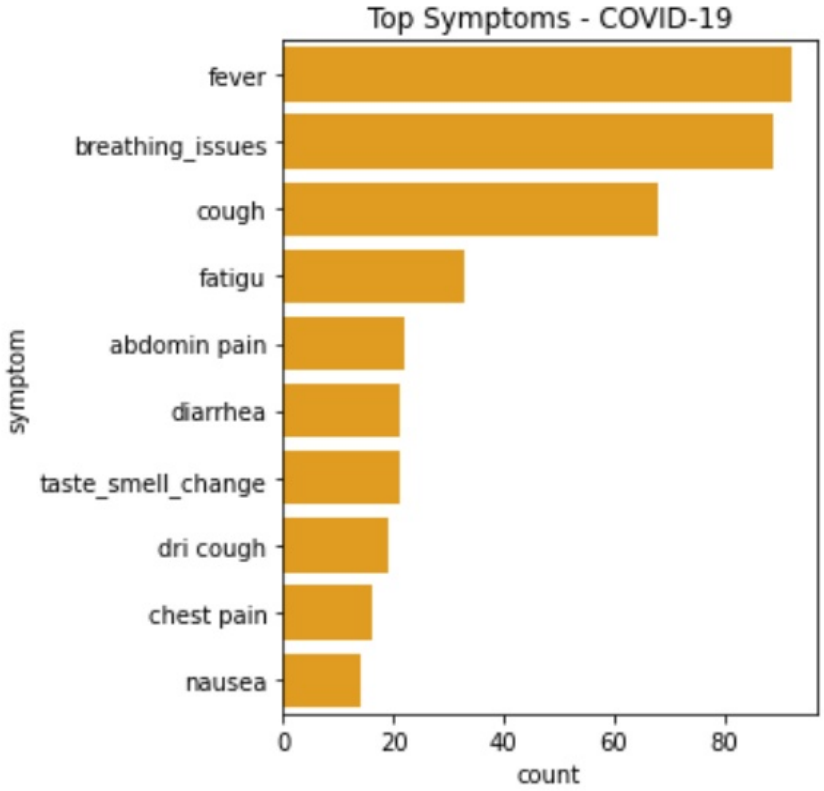
- 180 COVID-19 and 505 lymphoma case reports.
- 460 male, 196 female and 29 unidentified patients.
- Most patients in dataset were male
  - Similar male-to-female ratio for both diseases.
- Patients’ age varied between 0 and 100 years old
  - Lymphoma patients’ average age = 52
  - COVID-19 patients’ average age = 48.

Exploratory Analysis (cont.)



Exploratory Analysis (cont.)





Performance

Algorithm	Best Parameters	Best Performance (ROC AUC)
Catboost	'iterations': 100, 'l2_leaf_reg': 1, 'learning_rate': 0.04, 'subsample': 0.8, 'depth': 8	0.887
XGBoost	'colsample_bytree': 0.6, 'eta': 0.05, 'max_depth': 20, 'n_estimators': 50, 'subsample': 1.0	0.8895
LightGBM	'learning_rate': 0.05, 'max_depth': 5, 'min_child_samples': 5, 'num_leaves': 20, 'reg_alpha': 0.01	0.8898

Performance

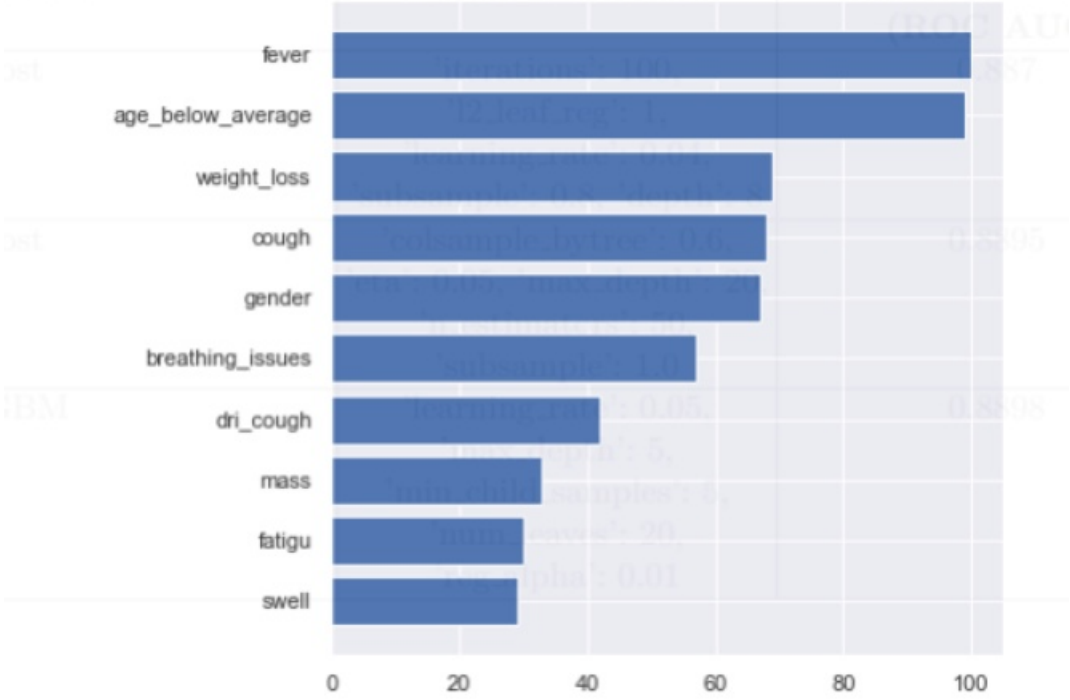


Fig. 2. Feature importance.

CONCLUSION

Conclusion

- With ROC AUC = 0.89, LightGBM model could be good tool for picking up lymphoma patients
- Benefits:
  - potentially reduce delay in lymphoma diagnosis
  - improve prognosis for lymphoma patients.
- Main limitations:
  - limited size of data set
  - Data imbalance
  - Need to try more hyperparameter combination