One Year Life Expectancy Post Thoracic Surgery Using IBM Watson

INTRODUCTION:

Cancer is a disease in which cells in the body grow out of control and is one of the most serious health problems in the world. Among various different types of cancer, lung cancer is one of the leading causes of death in both men and women. According to World Health Organization, it was observed that in the year 2018 2.09 million cases of lung cancer was registered and a total of 1.76 million died due to lung cancer. One of the reasons for the high death rate due to lung cancer is the late detection of the lung cancer. Also, the treatment and the prognosis depend on the type of the lung cancer, the stage and the patient's performance. Once the lung cancer is detected, possible treatments include thoracic surgery, chemotherapy and radiotherapy.

At present, there are few biological or technical methods to prevent cancer. Therefore, the project focuses on the early detection of the lung cancer which is the vital factor in the treatment which will in turn increase the survival rate post the treatment. Also, if lung cancer is detected, an attempt is made to predict the survival of the patient for a minimum span of one-year Post Thoracic Surgery which will assist the doctors to take the proper decision for the medication. The project develops a system that detects the lung cancer by processing the Computed Tomography (CT) image of the lungs under test. The system uses Convolution Neural Network (CNN) which takes CT scan lung images which is present in Digital Imaging and Communications in Medicine (DICOM) format as input and further outputs whether the image is cancerous or non-cancerous. If lung cancer is detected, further the life expectancy post thoracic surgery can be predicted using the Machine Learning techniques. Thus, the system can be taken as an aid for the doctors to effectively make decisions for the treatment of lung cancer patients.

1.1OVERVIEW:

Lung Cancer is a disease characterized by the uncontrolled cell growth in tissues of the lungs. It is one of the dangerous and life taking disease in the world. Early detection of lung cancer helps in identification of the treatment to be given which in turn increases the chances of survival of the lung cancer infected patient. This paper presents an approach which utilizes Convolution Neural network (CNN) to classify the CT lung images as cancerous or non-cancerous. The accuracy obtained by means of CNN is 95% which is more efficient when compared to accuracy obtained by the traditional neural network systems. Once the cancer is detected the paper also presents an approach to predict the

life expectancy of the lung cancer infected patient post thoracic surgery. The prediction part is implemented using Machine Learning techniques. Linear Discriminant Analysis gave an accuracy of 83.76% with a F-measure of 0.83 which is more compared to other machine learning algorithms used.

1.2Purpose:

Through this repository we can:

1.Apply the fundamental concepts of machine learning from an available dataset

2.Evaluate and interpret my results and justify my interpretation based on observed data set

3.Create notebooks that serve as computational records and document my thought process.\

The analysis is divided into four sec ons, saved in juypter notebooks in this repository

1. Identifying the problem and Data Sources
2. Exploratory Data Analysis
3. Pre-Processing the Data
4. Build model

Literature Survey:

EXISTING PROBLEM:

1:Patients who receive thoracic surgery for lung cancer do so with the expectation that their lives will be prolonged for a sufficient amount of time afterwards.

2.The problem to solve is whether there is a way to determine postoperative 1 year survival of lung cancer patients utilizing the patient attributes in the data set.

PROPOSED SOLUTION:

Disha Sharma *et al.* (2011), proposed an approach for the early detection of lung cancer by analyzing lungs CT images using Image Processing techniques[1]. The authors used bit-plane slicing, erosion and Weiner filter image processing techniques to extract the lung regions from the CT image. Further the extracted lung regions were segmented using Region growing segmentation algorithm and later Rule based model was used to detect the cancerous nodules. With the help of diagnostics indicator, it was observed that the proposed method achieved an overall accuracy of 80%.

Hamid Bagherieh *et al.* (2013) gave a methodology to detect and classify the lung nodules using Image processing and Decision- Making techniques[2]. Initially, image preprocessing was carried out on a CT images by using contrast enhancement and linear filtering. Next, the filtered image was segmented using Region growing Segmentation process. Further the features like area and color was given as input to the Fuzzy system which employed fuzzy membership function to find the abnormalities.

Sindhu V *et al*. (2014), proposed an approach where the authors aimed to classify the survival of lung cancer patients post thoracic surgery[3]. The authors used Naïve Bayes, PART, J48, OneR, Random Forest and Decision stump algorithm techniques to classify the target function. The performance measurement shows that Random Forest gave high accuracy of 95.65% compared to other ML techniques used.

Prashant Naresh *et al.* (2014), proposed a methodology to detect the lung cancer using Image processing and Neural Techniques[4]. Initially the CT image of lung was filtered to remove Gaussian white noise and Otsu's threshold technique was used to do the segmentation of the image. The structural and features were extracted and these features were given as input to the classifier. The SVM and ANN techniques were used for the classification and it was found that SVM techniques gave a higher accuracy of 95.12%.

Kwetishe Joro Danjuma (2015), proposed a methodology to predict the one- year survival of the patient post thoracic surgery[5]. Naïve Bayes, J48 and Multilayer

Perceptron algorithms were used to classify the target class. The Naïve Bayes gave an accuracy of 74.4%, J48 gave an accuracy of 81.8% and MLP gave an accuracy of 82.4%.

Abeer S Desuky *et al.* (2016), proposed a methodology to predict the post-operative life expectancy after Thoracic Surgery using attribute and selection ranking by using Simple Logistic, Multilayer Perceptron and J48 techniques[6]. The Simple Logistic gave an accuracy of 84.68%, MLP gave an accuracy of 81.28% and J48 gave an accuracy of 84.47%.

Peyman Rezaei *et al.* (2017), proposed a methodology to predict the one-year survival of lung cancer patients post thoracic surgery using the combination of Bayesian Network (BN) and Uniform counts discretization[7]. Both BN and Uniform counts discretization yielded an accuracy of 91.28%.

Wafa Alakwaa *et al.* (2017), demonstrated a CAD system for lung cancer classification of CT scans with unmarked nodules[8]. Thresholding was used as initial segment approach which produced best lung segmentation. A modified U-Net trained on LUNA16 data was used to detect nodule candidates. The U-Net output were fed into 3D Convolutional Neural Networks (CNNS) to ultimately classify CT scans as positive or negative for lung cancer. The 3D CNNS produced a test set accuracy of 86.6%.

Prediction of Life Expectancy

This is the second part of the system which aims at predicting the survival of lung cancer infected patient post thoracic surgery.

The dataset includes 17 attributes which are specified in Table- 1. Among all those attributes, Risk1Y is the target class specifying zero if the patient survives for at least one-year post thoracic surgery and one for those who died before completing one-year post surgery.

The visualization of the dataset is done using the Matplotlib and Seaborn libraries of Python. Further the essential attributes are found based on the Information Gain (IG) attribute evaluation which is used to find the importance of an attribute by using the Information Gain with respect to the target class.

IG (Class, Attribute) = E (Class) – E (Class | Attribute)

where, E stands for Entropy

After the IG Attribute Evaluation on all the 16 independent attributes in the dataset, it is found that the

attributes PRE19 and PRE32 gives the IG value as zero and hence are the least useful attributes for training the model. Therefore, these two attributes are eliminated and the remaining 14 attributes are used to train the model.

Dataset Attributes

Name	Description	Type
DGN	Diagnosis – specific combination of ICD-10 codes.	Nominal
PRE4	Forced Vital Capacity – FVC	Numeric
PRE5	Forced Expiratory Volume – FEV!	Numeric
PRE6	Performance Status – Zubrod Scale	Nominal
PRE7	Pain before Surgery	Binary
PRE8	Haemoptysis before Surgery	Binary
PRE9	Dyspnoea before surgery	Binary
PRE10	Coughing up blood before surgery	Binary
PRE11	Weakness before Surgery	Binary
PRE14	T in clinical TNM-size of original tumor, From OC11 (smallest) to	Nominal
	OC14(largest)	
PRE17	Type 2 DM – Diabetes Mellitus	Binary

PRE19	MI up to 6 months	Binary
PRE25	PAD – Peripheral Arterial Disease	Binary
PRE30	Smoking	Binary
PRE32	Asthma	Binary
AGE	Age at the time of Surgery	Numeric
RISK1Y	1 Year Survival (T-Died, F- Alive)	Binary

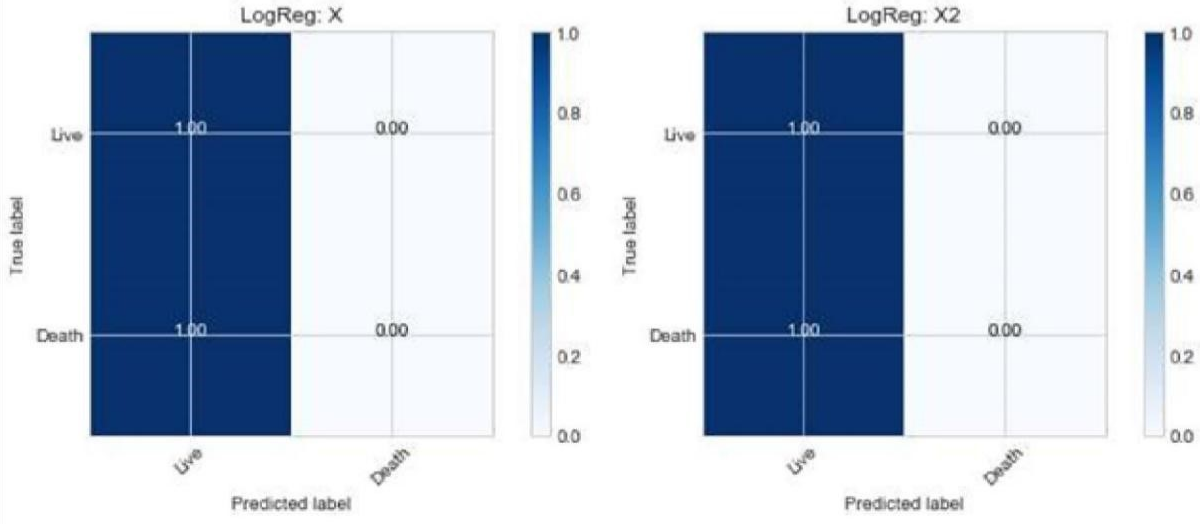
THEORITICAL ANALYSIS:

Machine Learning (Supervised Classification)

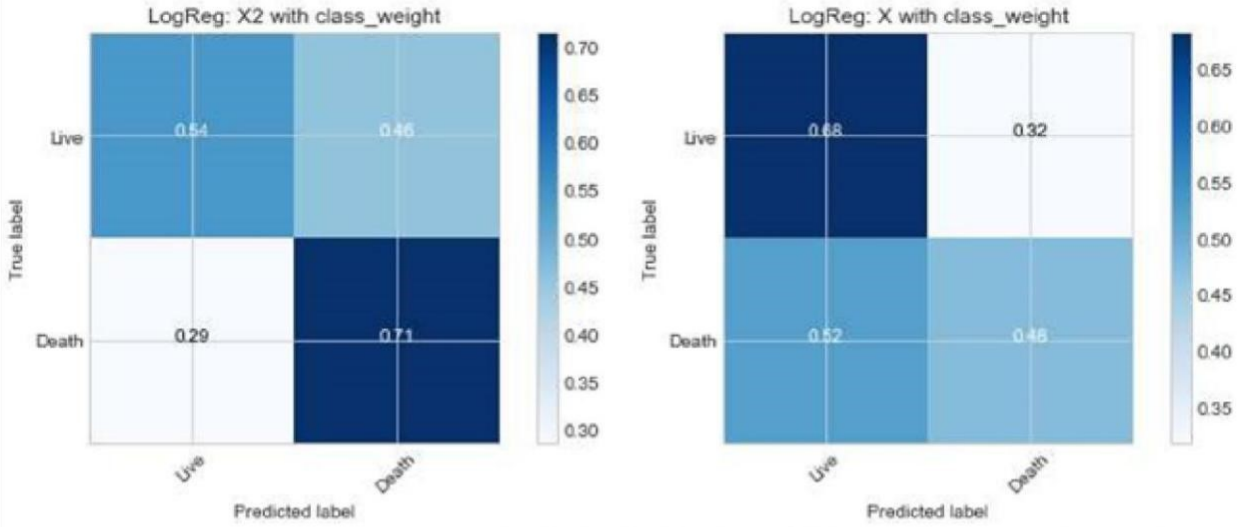
I focused on utilizing Logistic Regression and Random Forest Classifier for this supervised classification problem. From EDA and hypothesis testing to gather p values, I realized which attributes are significant in identifying the mean difference of those who lived and died in the 1 year period after surgery. I wanted focus the test on two different X data sets. The first data set drops the target variable, Death_1yr, and also the two attributes that shows little representation in the data itself, MI_6mo and Asthma. This data is referred to as X. The other data set only includes the attributes of significance concluded from the hypothesis testing in the EDA section: Performance, Dyspnoea, Cough, Tumor_Size, Diabetes_Mellitus. This data set is referred to as X2.

Since the data set is imbalanced and mostly live patients (85%), just predicting all live patients will give a high accuracy score ~85%. So for the model, accuracy will not be a good score method and instead I will look at average precision score, which summarizes the precision-recall curve. Also for the imbalance, there are couple options including downsampling, upsampling, or adjusting class weights to balance the classes. Since downsampling will create a small data set to work with and upsampling may complicate the data further, I will focus on adjusting the class weights.

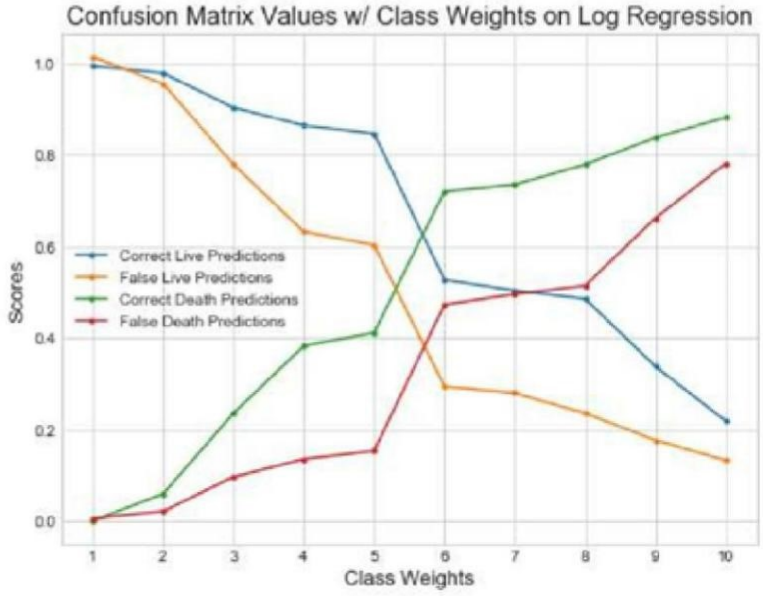
Logistic Regression



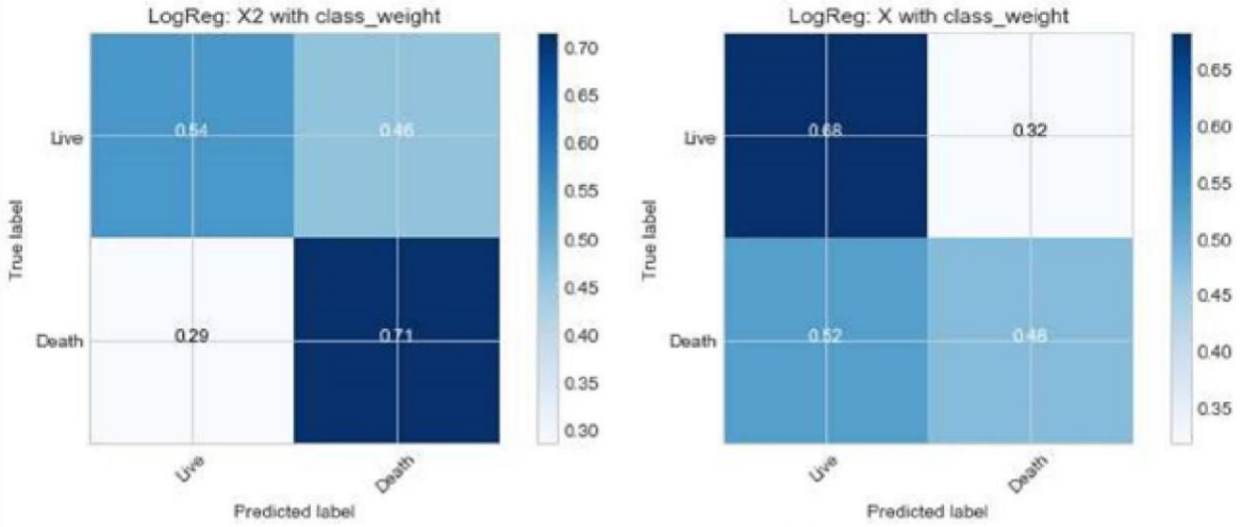
Since the data set is imbalanced with only 15% patient death, the results of the model without any class weight to offset this imbalance favors the live column in the confusion matrix. As you can see above, the model predicts mostly all live patients to maximize the accuracy score to 85%, the size of the live patient data, in both the X and X2 data sets.



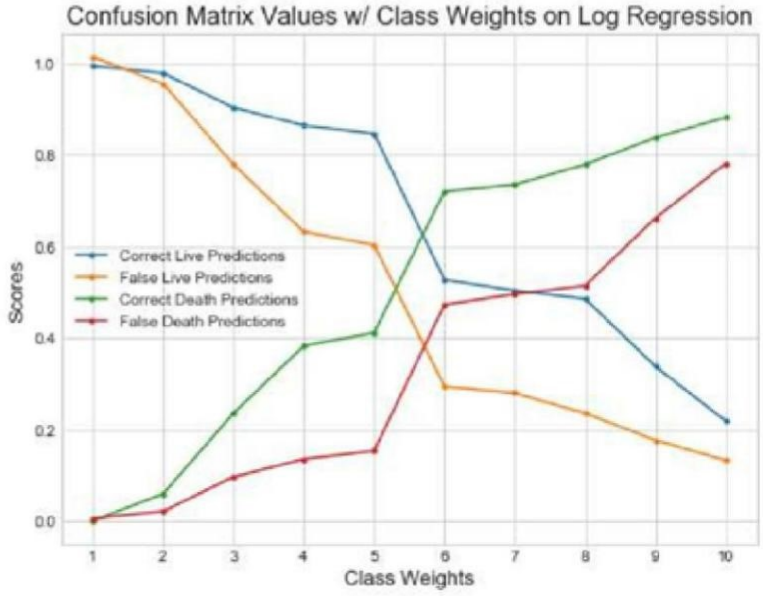
With the class weight parameter, the death prediction rate increases at the cost of live patient prediction, and also the accuracy. In order to see the effectiveness of the model for my purpose, the confusion matrix or classification report can be used to assess the death predictions. Also, the average precision score is a good summary of the precision-recall curve, which is useful in this case.



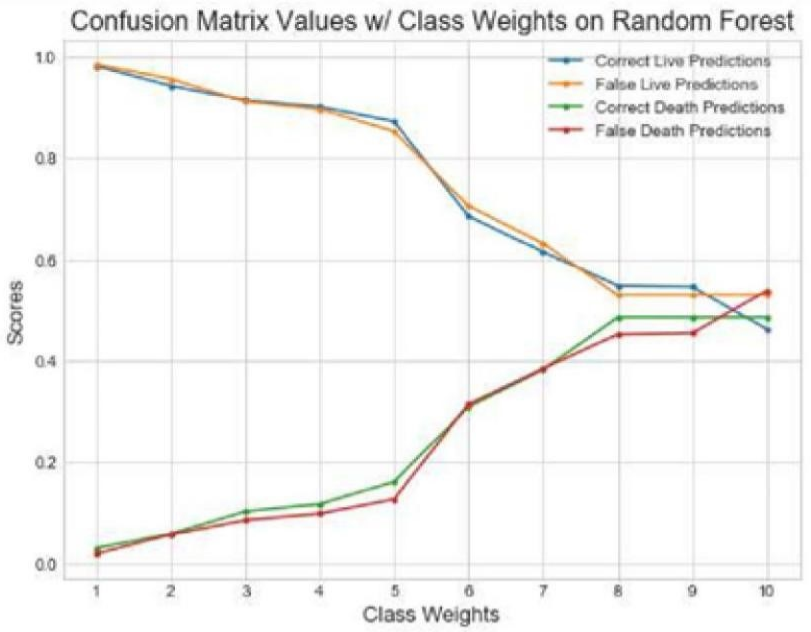
Since the data set is imbalanced with only 15% patient death, the results of the model without any class weight to offset this imbalance favors the live column in the confusion matrix. As you can see above, the model predicts mostly all live patients to maximize the accuracy score to 85%, the size of the live patient data, in both the X and X2 data sets.

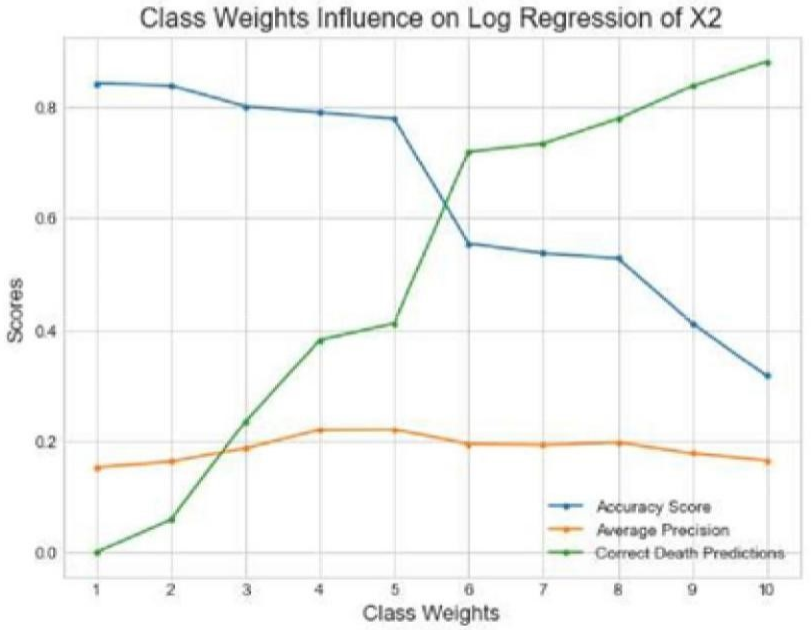


With the class weight parameter, the death prediction rate increases at the cost of live patient prediction, and also the accuracy. In order to see the effectiveness of the model for my purpose, the confusion matrix or classification report can be used to assess the death predictions. Also, the average precision score is a good summary of the precision-recall curve, which is useful in this case.



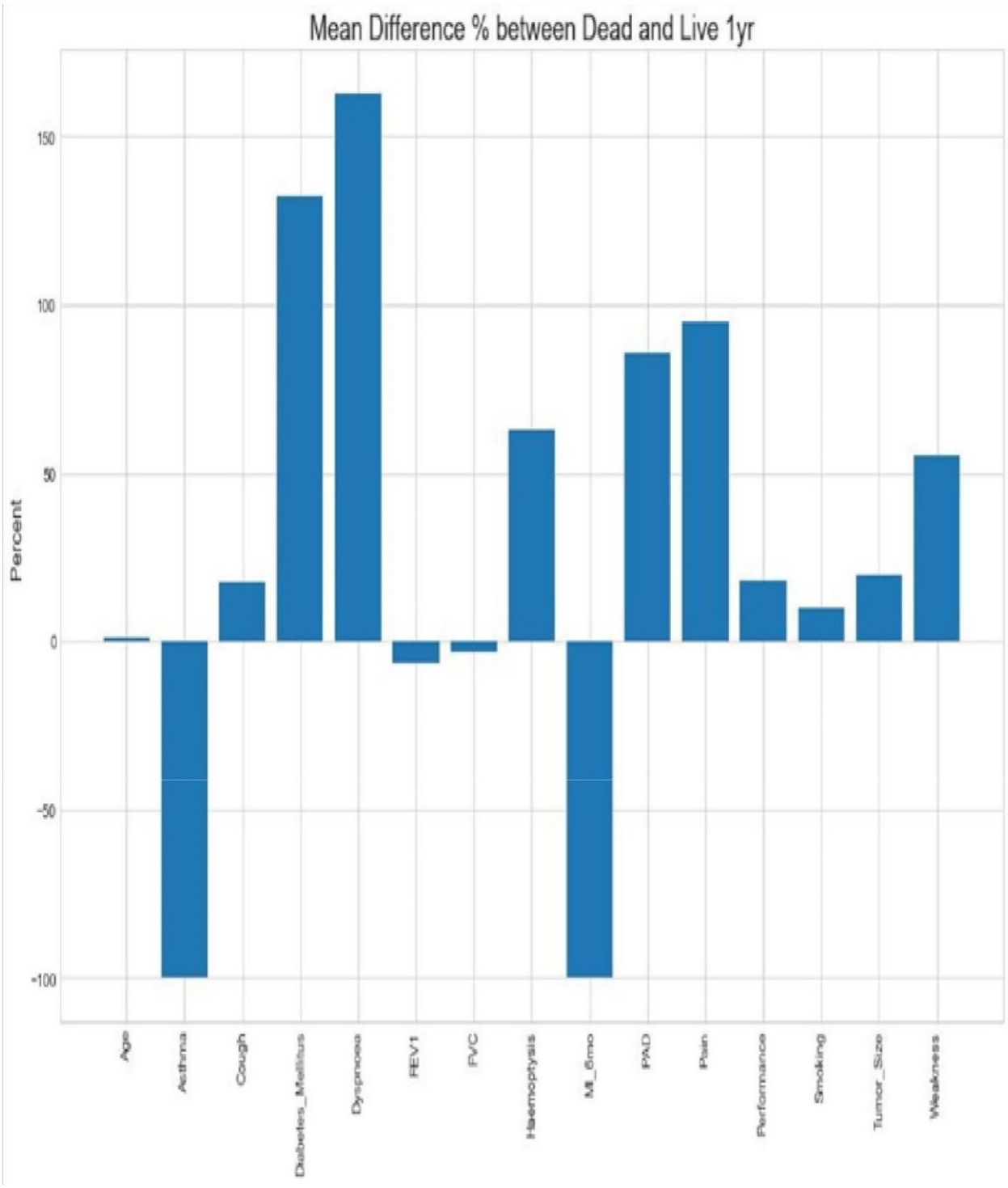
Similar to the Logistic Regression models, the Random Forest predicts deaths better with a class weight parameter to balance the data. The plot reveals the cost of correct live predictions and benefit of correct death predictions with differing class weights. It is interesting to note the different pattern this model takes compared to the log regression graphs above. Based solely on average precision, the log regression produces better results.

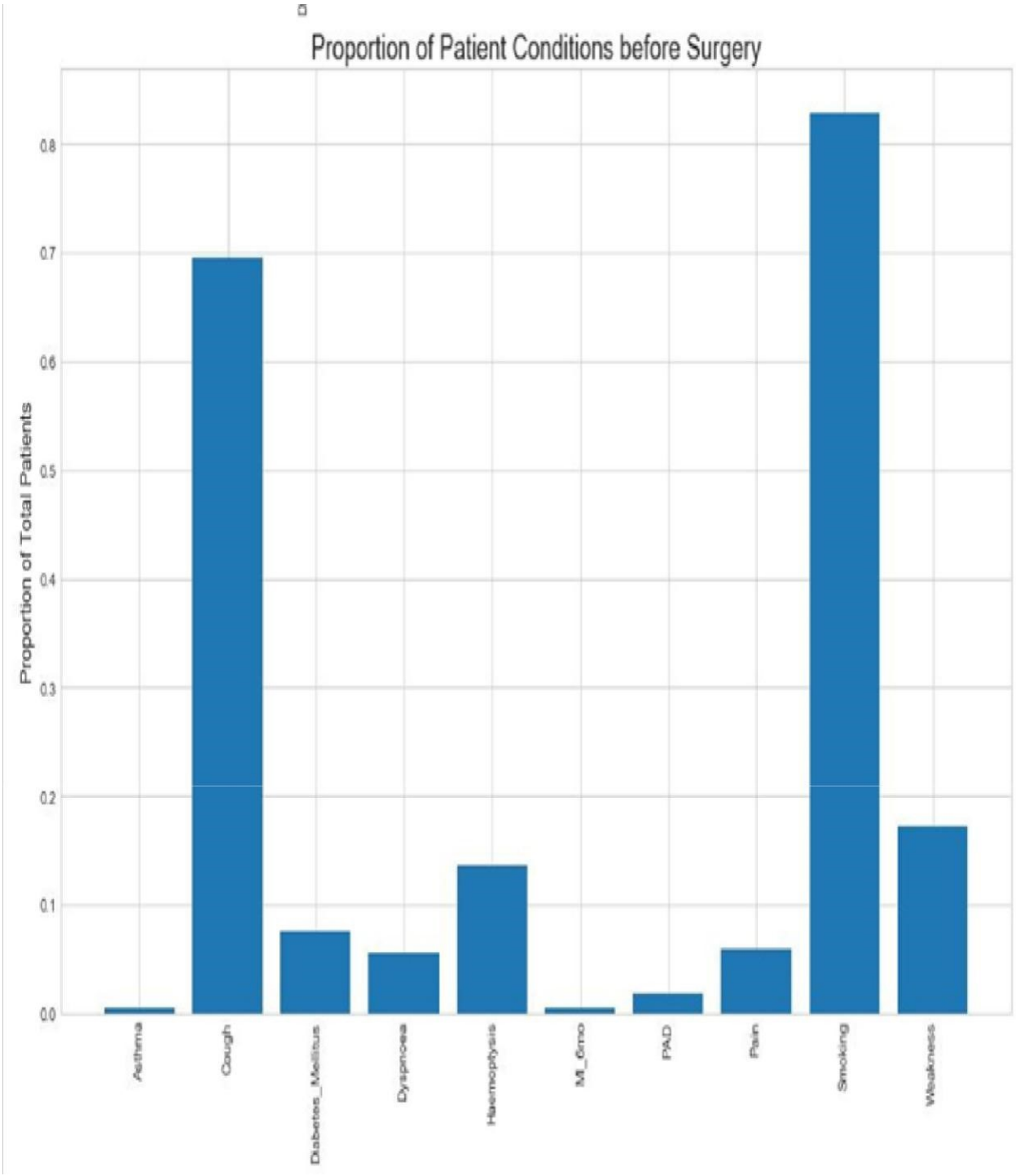




However, with hyperparameter tuning, the random forest classifier delivers higher average precision scores compared to what the log regression model did in any class weight value. It is interesting to note that GridSearchCV model notes the best parameter as having no class weight argument with the tested parameters in the report. So, there probably are combinations of hyperparameters that perform better than the models highlighted above with

RESULT:


Mean Difference % between Dead and Live 1yr
Percent
150
100
50
0
-50
-100
Age
Asthma
Cough
Diabetes_Mellitus
Dyspnoea
FEV1
FVC
Haemoptysis
MI_6mo
PAD
Pain
Performance
Smoking
Tumor_Size
Weakness


Proportion of Patient Conditions before Surgery
Proportion of Total Patients
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Asthma
Cough
Diabetes_Mellitus
Dyspnoea
Haemoptysis
MI_6mo
PAD
Pain
Smoking
Weakness

Advantages	Extensive perioperative analgesia	Easy and fast to perform
		Hemodynamic stability
		Operating room efficiency
Disadvantages	Time consuming	Limited perioperative analgesia
	Skill demanding	
	Risks of neurological injury	

INB = intercostal nerve block, TEA = thoracic epidural anesthesia.

APPLICATIONS OF THOARAIC SURGERY:

Lymphatic mapping:

Patients with Stage I lung cancer have a recurrence rate of nearly 40% and the overall 5-year survival rate is just 52% [12]. The high rate of disease recurrence suggests that these patients are frequently understaged and, subsequently, undertreated. By identifying lymph nodes at highest risk for tumour metastasis, SLN mapping helps identify patients who could benefit from adjuvant therapy. Several NP-based molecular imaging techniques have been explored for potential in the mapping of tumour-draining lymph nodes in human cancers [13, 14]. Quantum dots (QDs) have been utilized as lymphatic mapping agents in a porcine model of non-small-cell lung cancer (NSCLC), with favourable properties including appropriate size for lymphatic migration and visible histological fluorescence (Fig. 1) [15]. More recently, superparamagnetic iron oxide (SPIO) NPs were coupled with magnetic resonance imaging (MRI) to create

ultrasensitive nanoprobes for cellular and molecular imaging of various cancers. Notably, SPIO-enhanced MRI technology has been shown to detect tumour metastases in the SLNs of breast, bladder and prostate cancer patients [16, 17]. In a 2011 study of 102 breast cancer patients with clinically negative axillary nodes, Motomura *et al* . [16] demonstrated that SPIO-enhanced MRI achieved almost 90% accuracy in the detection of SLN metastases. These promising results warrant the

upcoming investigation of SPIO-enhanced MR imaging as an SLN mapping strategy for lung and oesophageal cancer patients.

Image-guided nanosurgery

NP-based technologies possess several properties that enhance imaging of biological targets, including the ability to amplify contrast signal, unique physiochemical characteristics such as magnetic, thermal and pH-responsive phase changes, and the ability to modify pharmacokinetics via alterations in surface chemistry [10]. The development of targeted near-infrared fluorescent (NIRF) and surface-enhanced Raman scattering imaging probes for the evaluation of surgical margins and the real-time intraoperative identification of residual disease is a major focus of recent investigation. These technological advances have been translated to patient care with excellent initial results. Currently, NIRF-image-guided surgery utilizing 5-aminolevulinic acid (5-ALA) generates tumour fluorescence and better tumour visualization in patients with malignant glioma. 5-ALA precursor in the haemoglobin synthesis pathway elicits the accumulation of fluorescent porphyrins in various epithelia and cancerous tissues. Oral administration of 5-ALA leads to preferential accumulation of fluorescent porphyrins within glioma tissue, enabling more complete oncological resection and improved progression-free survival [24]. Ongoing research efforts are focused on the clinical translation of various Raman NPs that can be specifically utilized for NIRF imaging of various solid tumours, including lung cancer. It is anticipated that this technology will

improve intraoperative guidance of surgical resection with negative margins in the future.

Several investigators have developed triple-modality NP imaging probes for use with a number of cancer imaging modalities, including PET, NIRF and MRI, to allow for preoperative tumour localization and surgical planning. One of the best characterized strategies utilizes MRI-photoacoustic silica-coated Raman NPs [25]. Kircher *et al* . reported the intravenous injection of Raman NPs into glioblastoma-bearing mice, which leads to intratumoural accumulation and retention for several days. Triple-modality MRI-based photoacoustic imaging was then used to guide intraoperative resection and accurately delineate tumour margins (Fig. 2) [25]. Ultra-sensitive Raman imaging can be utilized to detect and remove residual microscopic tumour burden. Coupling of high-resolution tissue-penetrating photoacoustic imaging and tumour-homing Raman NPs has potential to enable localization and accurate resection of non-palpable deep parenchymal lung lesions. Such approaches highlight the potential for translation of emerging nanotechnology platforms to thoracic surgery, particularly with the focus of enabling more accurate tumour imaging and improve oncological resection in patients with thoracic cancers

CONCLUSION :

Detection of lung cancer is one of the challenging problems in medical field due to structure of cancer cells, where most of the cells are overlapped to each other. Detection of lung cancer in the early stage is curable. The system contains two parts. One is Lung Cancer Detection part and the other is the Prediction of Life Expectancy Post Thoracic Surgery. Both the parts can run independently. The system is provided as a web application where anyone can upload a CT scan image of the lung in the front end and check out whether that image is infected with Lung Cancer. At the backend the image uploaded is preprocessed, segmented and predicted using the CNN model which is already trained. Also, the system provides a form requesting for

the 14 attributes required to predict the survival span post Thoracic Surgery. Once the form is submitted, the form inputs are run on the LDA model and a response of whether the patient will survive or not is produced. CNN model used to detect lung cancer gave an accuracy of 95% and the LDA classifier gave the highest accuracy of 83.76% compared to other 3 algorithms used. The system considers only CT lung images as input, but further the system can be enhanced to take MRI (Magnetic Resonance Imaging) or PET (Position Emission Tomography) as input. Also, in the prediction of survival part, higher accuracies of the classifier can be achieved by considering large amount of dataset.

FUTURE SCOPE :

Thoracic surgery is living an enthusiastic era of thriving innovations. The changes in the fields of diagnostics, in the therapeutic options, the development of surgical techniques and the adoption of novel technologies has radically changed the horizons of our specialty. We must embrace the innovations, learn navigational bronchoscopy, understand the multiple targeted therapies, and learn new ways to operate on advanced cases. We will not be able to stay complacent with our current 3 ports video-assisted thoracoscopic surgery (VATS) technique with open instruments, therefore every surgeon needs to be watching for each latest development as it happens. In this brief manuscript we will summarize some of the most important development over the recent years and forecast how this will impact on the patients affected by thoracic malignancies. As thoracic surgeons we have to embraces these developments in order to redefine our specialty.

Bibliography

1. Hendrix RJ, Lee C, Friedrich AK, Rouanet E, Larkin AC, LaFemina J. Prophylactic VersusTherapeutic Mastectomy: A Contemporary Analysis of the ACS-NSQIP

Database.ClinBreastCancer. 2019 Jun;19(3):e428-e432.
2. Hendrix RJ, Martins PN, Stoff JS, Ahearn A, Bozorgzadeh A, Movahedi B. Successful RenalTransplantation after Presumed Cyanide Toxicity Treated with Hydroxocobalamin and Review ofthe Literature.CaseRep Transplant. 2018 Sep 9;2018:3753479.
3. McKie K, McLoughlin RJ, Hirsh MP, Cleary MA, Aidlen JT. Risk Factors for Venous Thromboembolism in Children and Young Adults With Inflammatory Bowel Disease.JSurg Res.2019 Jun 7;243:173-179.
4. McLoughlin RJ, Green J, Nazarey PP, Hirsh MP, Cleary M, Aidlen JT. The risk of snow sportinjury in pediatric patients.AmJ Emerg Med. 2019 Mar;37(3):439-443.
5. Hoang CM, Alavi K, Flahive JM, Sturrock PR, Maykel JA, Davids JS. Impact of the "WeekendEffect" for Hospital Discharges on Readmissions After Elective Colectomy.DisColon Rectum.2019 Apr;62(4):476-482.
6. Sarani B, Paspulati RM, Hambley J, Efron D, Martinez J, Perez A, Bowles-Cintron R, Yi F, Hill S, Meyer D, Maykel J, Attalla S, Kochman M, Steele S. A multidisciplinary approach to diagnosisand management of bowel obstruction.

1. Curr Probl Surg. 2018 Oct;55(10):394-438.

2. Davids JS, Lyu HG, Hoang CM, Daniel VT, Scully RE, Xu TY, Phatak UR, Damle A, MelnitchoukN. Female Representation and Implicit Gender Bias at the 2017 American Society of Colon andRectal Surgeons' Annual Scientific and Tripartite Meeting.DisColon Rectum. 2019Mar;62(3):357-362.
3. Min M, Noujaim MG, Green J, Schlieve CR, Vaze A, Cahan MA, Cave DR. Role of MucosalProtrusion Angle in Discriminating between True and False Masses of the Small Bowel on VideoCapsule Endoscopy.

4. J Clin Med. 2019 Mar 27;8(4). pii: E418.

5. Murray CR, Balsam LB. Commentary: Outliers-Salvage operations and the Society of Thoracic

Surgeons risk model. JThorac Cardiovasc Surg. 2019 Feb 27. pii: S0022-5223(19)30491
6. Shahi N, Arosemena M, Kwon J, DiMuzio P, Abai B, Salvatore DM. A rare case of*Clostridiumsepticum* aortitis with colon adenocarcinoma.JVasc Surg Cases

Innov Tech. 2018 Apr23;4(2):87-90.

7. Hendrix RJ, Damle A, Williams C, Harris A, Spanakis S, Lambert DH, Lambert LA. RestrictiveIntraoperative Fluid Therapy is Associated with Decreased Morbidity and Length of StayFollowing Hyperthermic Intraperitoneal Chemoperfusion.AnnSurg Oncol. 2019 Feb;26(2):490-496.

SOURCE CODE:

```
%matplotlibinline

        import pandas as pd  import

        numpy as np
        import matplotlib.pyplot as plt  import

        seaborn as sns sns.set_style("whitegrid")


In [2]:import typesimport
        pandas as pd
        from  botocore.client  import  Config
        import  ibm_boto3  def    __iter__(self):
        return 0
```

1. *@hidden_cell*
2. *The following code accesses a file in your IBM Cloud Object Storage. It incl udes your credentials.*
3. *You might want to remove those credentials before you share the notebook.*

```
client_ca664f4bd18b4ee7be1a254407d5a9cb =
ibm_boto3.client(service_name='s3',

    ibm_api_key_id='LnDsssRcQkVEuzSKFi8xiJ_iPDfljTBahv39
piIi9d8c',
ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token
", config=Config(signature_version='oauth'),

    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')
```

```
body=
client_ca664f4bd18b4ee7be1a254407d5a9cb.get_object(Bucket='oneyearlifeexpectancypostthoracic-donotdelete-pr-ydkg5pb794dfn6',Key='Dataset.csv')['Body' ]
```

1. *add missing __iter__ method, so pandas accepts body as file-like object* **if** **not** hasattr(body,"__iter__"): body.__iter__=types.MethodType(__iter__, body)

   Dataset = pd.read_csv(body) Dataset.head()

In [3]: Dataset . describe ()

Out[2]:

	Diagnosis	FVC	FEV1	Performance	Pain	Haemoptysis	Dyspnoea	Cough	Weakness	Tumo
0	2	2.88	2.16	1	0	0	0	1	1	
1	3	3.40	1.88	0	0	0	0	0	0	
2	3	2.76	2.08	1	0	0	0	1	0	
3	3	3.68	3.04	0	0	0	0	0	0	
4	3	2.44	0.96	2	0	1	0	1	1	

Out[3]:

	Diagnosis	FVC	FEV1	Performance	Pain	Haemoptysis	Dyspnoea	
count	454.000000	454.000000	454.000000	454.000000	454.000000	454.000000	454.000000	4
mean	3.092511	3.287952	2.516855	0.795154	0.059471	0.136564	0.055066	
std	0.715817	0.872347	0.771899	0.531459	0.236766	0.343765	0.228361	
min	1.000000	1.440000	0.960000	0.000000	0.000000	0.000000	0.000000	
25%	3.000000	2.600000	1.960000	0.000000	0.000000	0.000000	0.000000	
50%	3.000000	3.160000	2.360000	1.000000	0.000000	0.000000	0.000000	
75%	3.000000	3.840000	2.977500	1.000000	0.000000	0.000000	0.000000	
max	8.000000	6.300000	5.480000	2.000000	1.000000	1.000000	1.000000	

In [4]:
```
# Stats for live and death after 1 yr patientslive=
Dataset[Dataset['Death_1yr'] == 0] death =
Dataset[Dataset['Death_1yr'] == 1]

cond = ['FVC', 'FEV1', 'Performance', 'Pain', 'Haemoptysis', 'Dyspnoea', 'Coug h',
    'Weakness',\ 'Tumor_Size', 'Diabetes_Mellitus', 'MI_6mo', 'PAD', 'Smoking', 'Asthm
    a', 'Age']

l = [np.mean(live[c]) for c in cond] d =
[np.mean(death[c]) for c in cond]

ld = pd.DataFrame(data={'Attribute': cond, 'Live 1yr Mean': l, 'Death 1yr Mea n': d})

ld = ld.set_index('Attribute')

print('Death:{:d}, Live:{:d}'.format(len(death), len(live)))

print("1 year death:{:.2f}% out of 454
patients".format(np.mean(Dataset.Death_1yr)*100)) ld
```

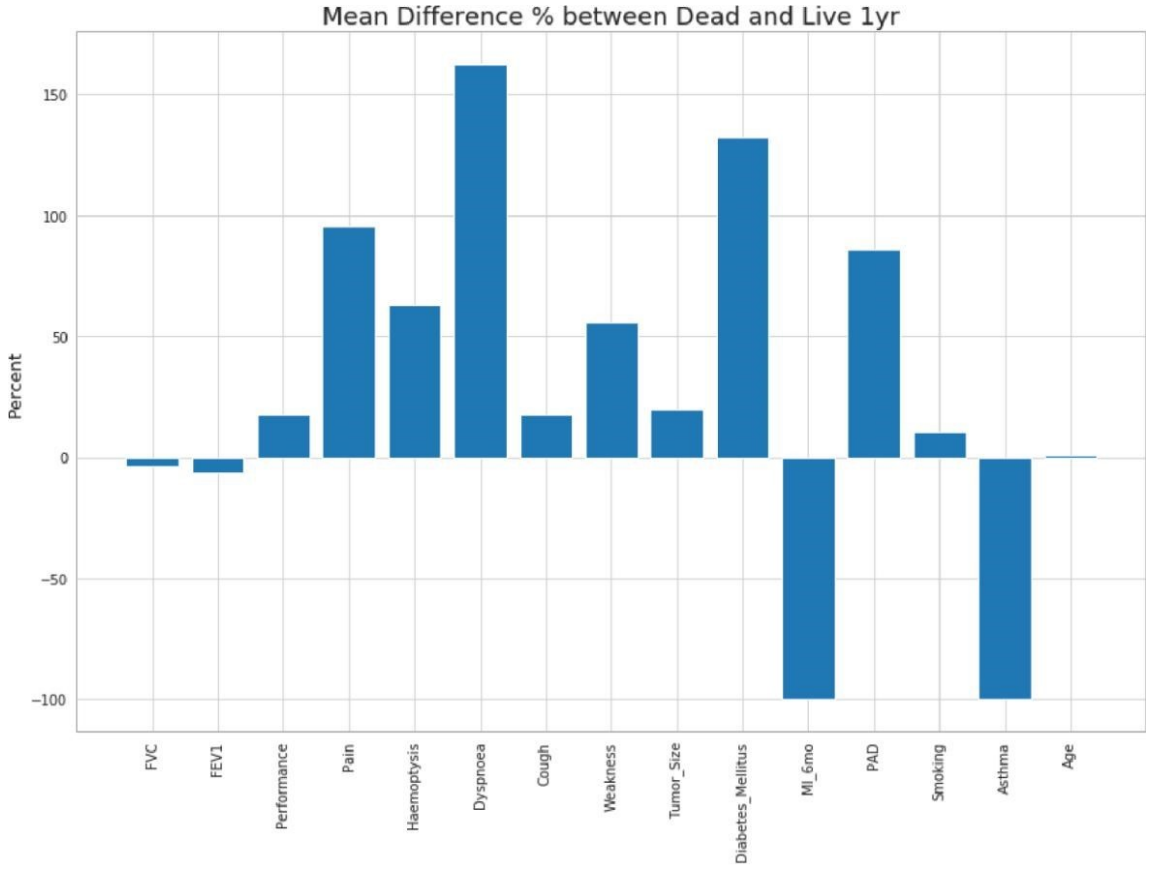
```
Death: 69, Live: 385

1 year death: 15.20% out of 454 patients
```

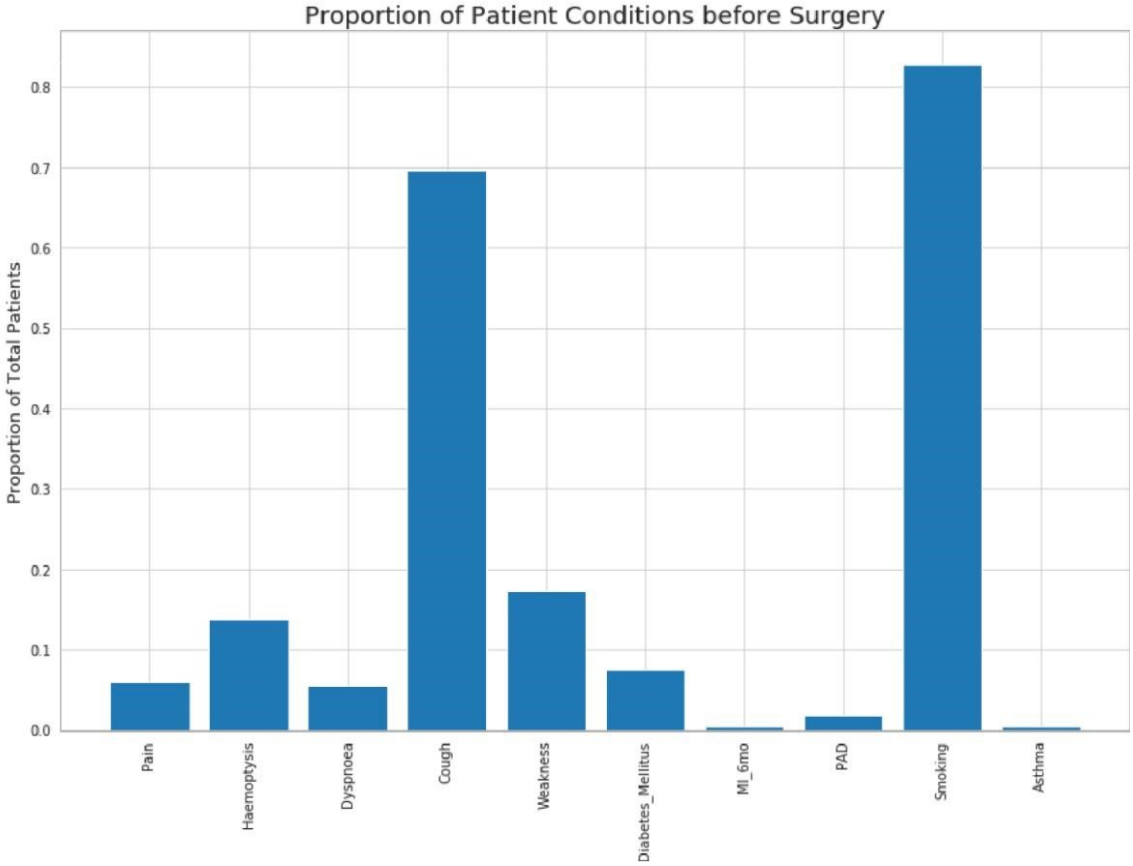
Out[4]:

Attribute	Live 1yr Mean	Death 1yr Mean
FVC	3.304597	3.195072
FEV1	2.540805	2.383188
Performance	0.774026	0.913043
Pain	0.051948	0.101449
Haemoptysis	0.124675	0.202899
Dyspnoea	0.044156	0.115942
Cough	0.677922	0.797101
Weakness	0.158442	0.246377
Tumor_Size	1.683117	2.014493
Diabetes_Mellitus	0.062338	0.144928
MI_6mo	0.005195	0.000000

PAD	0.015584	0.028986
Smoking	0.815584	0.898551
Asthma	0.005195	0.000000
Age	62.677922	63.333333

In [5]:
```
# Percentage difference in means of live vs death patients
d = np.array(d)
l = np.array(l)

p_diff = (d-l)/l*100

fig, axes = plt.subplots(2,1,figsize=(12,18))

axes[0].bar(cond, p_diff)
axes[0].set_title('Mean Difference % between Dead and Live 1yr', fontsize=18)
axes[0].set_xticks(cond)
axes[0].set_xticklabels(cond, rotation=90)
axes[0].set_ylabel('Percent', fontsize=13)

# Count plot of true/false condition columns

tf_col = ['Pain', 'Haemoptysis', 'Dyspnoea', 'Cough', 'Weakness', 'Diabetes_Me
llitus', 'MI_6mo', 'PAD', 'Smoking', 'Asthma']
tf_sum = [Dataset[col].sum()/454 for col in tf_col]

axes[1].bar(tf_col, tf_sum)
axes[1].set_xticks(tf_col)
axes[1].set_xticklabels(tf_col, rotation=90)
axes[1].set_ylabel('Proportion of Total Patients', fontsize=13)
axes[1].set_title('Proportion of Patient Conditions before Surgery', fontsize=
18)

plt.tight_layout()

plt.show()
```


Mean Difference % between Dead and Live 1yr
Percent
150
100
50
0
−50
−100
FVC
FEV1
Performance
Pain
Haemoptysis
Dyspnoea
Cough
Weakness
Tumor_Size
Diabetes_Mellitus
MI_6mo
PAD
Smoking
Asthma
Age


Proportion of Patient Conditions before Surgery
Proportion of Total Patients
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Pain
Haemoptysis
Dyspnoea
Cough
Weakness
Diabetes_Mellitus
MI_6mo
PAD
Smoking
Asthma

```
#plots of Diagnosis, Tumor_Size, Performance with difference of live and death data

fig, axes = plt.subplots(3,1,figsize=(10,15))

sns.countplot(x='Diagnosis', hue='Death_1yr', data=df, palette='Blues_d',
ax=axes[0]).set_title('Diagnosis', fontsize=18)
sns.countplot(x='Tumor_Size', hue='Death_1yr', data=df, palette='Blues_d',
ax=axes[1]).set_title('Tumor_Size', fontsize=18)
sns.countplot(x='Performance', hue='Death_1yr', data=df, palette='Blues_d',
ax=axes[2]).set_title('Performance', fontsize=18)

plt.tight_layout()

defpermutation_sample(data1, data2):
    """Generate a permutation sample from two data sets."""
    data = np.concatenate((data1, data2))
    permuted_data = np.random.permutation(data)

    perm_sample_1 = permuted_data[:len(data1)]
    perm_sample_2 = permuted_data[len(data1):]

    return perm_sample_1, perm_sample_2

defdraw_perm_reps(data_1, data_2, func, size=1):
    """Generate multiple permutation replicates."""
    perm_replicates = np.empty(size)

    for i inrange(size):
        perm_sample_1, perm_sample_2 = permutation_sample(data_1, data_2)
        perm_replicates[i] = func(perm_sample_1, perm_sample_2)

    return perm_replicates

defdiff_of_means(data_1, data_2):
    """Difference in means of two arrays."""
    diff = np.mean(data_1) - np.mean(data_2)
    return diff
# Hypothesis testing with Permutations of data
condition = ['FVC', 'FEV1', 'Performance', 'Pain', 'Haemoptysis', 'Dyspnoea', 'Cough', 'Weakness',\
            'Tumor_Size', 'Diabetes_Mellitus', 'MI_6mo', 'PAD', 'Smoking', 'Asthma', 'Age']
p_val = []
```

```
for c in condition:
    empirical_diff_means = diff_of_means(death[c], live[c])
    perm_replicates = draw_perm_reps(death[c], live[c], diff_of_means, size=10000)
    if empirical_diff_means >0:
        p = np.sum(perm_replicates >= empirical_diff_means) /len(perm_replicates)
        p_val.append(p)
    else:
        p = np.sum(perm_replicates <= empirical_diff_means) /len(perm_replicates)
        p_val.append(p)

print(list(zip(condition, p_val)))
```

```
fromsklearn.linear_modelimport LogisticRegression
fromsklearn.model_selectionimport cross_val_score, cross_val_predict, train_test_split,
GridSearchCV
fromsklearn.metricsimport confusion_matrix, classification_report, accuracy_score,
average_precision_score
fromsklearn.ensembleimport RandomForestClassifier
importitertools
```

In [21]:

```
# All attributes excluding target variable, Asthma, and MI_6mo
X = df.drop(['Death_1yr', 'MI_6mo', 'Asthma'], axis=1)

# Attributes of Significance from Hypothesis Testing
X2 = df[['Performance', 'Dyspnoea', 'Cough', 'Tumor_Size', 'Diabetes_Mellitus']]
y = df['Death_1yr']
```

In [44]:

```
defplot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
```

```
    print('Confusion matrix')

  print(cm)

  plt.imshow(cm, interpolation='nearest', cmap=cmap)
  plt.title(title)
  plt.colorbar()
  tick_marks = np.arange(len(classes))
  plt.xticks(tick_marks, classes, rotation=45)
  plt.yticks(tick_marks, classes)

  fmt ='.2f'if normalize else'd'
  thresh = cm.max() /2.
  for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
          horizontalalignment="center",
          color="white"if cm[i, j] > thresh else"black")

  plt.tight_layout()
  plt.ylabel('True label')
  plt.xlabel('Predicted label')
```

In [49]:

```
defmodel_report(model, X, y, title, weight=None):
  """Takes in classifier model with X data and class weight to display scores and confusion matrix."""

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=1111, stratify=y)

  clf = model(class_weight=weight, random_state=1111)

  clf.fit(X_train, y_train)
  y_pred = clf.predict(X_test)

  class_names = ['Live', 'Death']

  print('Accuracy: {:.2f}'.format(accuracy_score(y_test, y_pred)))
  print('Average Precision: {:.2f}'.format(average_precision_score(y_test, y_pred)))
  print(classification_report(y_test, y_pred, target_names=class_names))

  cnf_matrix = confusion_matrix(y_test, y_pred)
  np.set_printoptions(precision=2)

  # Plot confusion matrix
```

```
    plt.figure()
    plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True, title=title)
    plt.show()
```

In [37]:

```
def class_weights_plot(model, title):
    """Plots accuracy, average precision, and correct death predictions of different class weights for X2 data."""
    class_weights = [1,2,3,4,5,6,7,8,9,10]
    acc_scores = []
    prec_scores = []
    correct_deaths = []

    for cw in class_weights:
        clf = model(class_weight={0: 1, 1: cw}, random_state=1111)
        predicted = cross_val_predict(clf, X2, y, cv=5)

        acc_scores.append(accuracy_score(y, predicted))
        prec_scores.append(average_precision_score(y, predicted))
        correct_deaths.append(confusion_matrix(y, predicted)[1,1]/68)

    plt.figure(figsize=(8,6))
    plt.plot(class_weights, acc_scores, marker='.', label='Accuracy Score')
    plt.plot(class_weights, prec_scores, marker='.', label='Average Precision')
    plt.plot(class_weights, correct_deaths, marker='.', label='Correct Death Predictions')
    plt.xticks(class_weights)
    plt.xlabel('Class Weights', fontsize=13)
    plt.ylabel('Scores', fontsize=13)
    plt.legend()
    plt.title(title, fontsize=16)

    plt.show()
```

In [35]:

```
def class_weights_tf_plot(model, title):
    """Plots confusion matrix values of different class weights for X2 data."""
    class_weights = [1,2,3,4,5,6,7,8,9,10]
    true_live = []
    false_live = []
    true_death = []
    false_death = []

    for cw in class_weights:
        clf = model(class_weight={0: 1, 1: cw}, random_state=1111)
```

```
        predicted = cross_val_predict(clf, X2, y, cv=5)

        true_live.append(confusion_matrix(y, predicted)[0,0]/385)
        false_live.append(confusion_matrix(y, predicted)[1,0]/68)
        true_death.append(confusion_matrix(y, predicted)[1,1]/68)
        false_death.append(confusion_matrix(y, predicted)[0,1]/385)

    plt.figure(figsize=(8,6))
    plt.plot(class_weights, true_live, marker='.', label='Correct Live Predictions')
    plt.plot(class_weights, false_live, marker='.', label='False Live Predictions')
    plt.plot(class_weights, true_death, marker='.', label='Correct Death Predictions')
    plt.plot(class_weights, false_death, marker='.', label='False Death Predictions')
    plt.xticks(class_weights)
    plt.xlabel('Class Weights', fontsize=13)
    plt.ylabel('Scores', fontsize=13)
    plt.legend()
    plt.title(title, fontsize=16)

    plt.show()

# Log Reg on X with no class weights
model_report(LogisticRegression, X, y,'LogReg: X')

# Log Reg on X with class weight balanced since imbalanced death numbers (15%)
model_report(LogisticRegression, X, y,'LogReg: X with class_weight','balanced')
# X2 log reg with no class weights
model_report(LogisticRegression, X2, y,'LogReg: X2')
# X2 log reg with class weight balanced
model_report(LogisticRegression, X2, y,'LogReg: X2 with class_weight','balanced')

# Plot different class weights influence on Log Reg X2
class_weights_tf_plot(LogisticRegression, 'Confusion Matrix Values w/ Class Weights on Log
Regression')
# Plot different class weights influence on LogReg X2
class_weights_plot(LogisticRegression, 'Class Weights Influence on Log Regression of X2')
# X2 Random Forest Classifier with equal class weights 1:1
model_report(RandomForestClassifier, X2, y,'RF: X2')
# X2 Random Forest Classifier with class weight 5.67
model_report(RandomForestClassifier, X2, y, 'RF: X2 with class_weight','balanced')
# Plot different class weights influence on RF classifier
class_weights_tf_plot(RandomForestClassifier, 'Confusion Matrix Values w/ Class Weights on
Random Forest')
```

```
# Plot different class weights influence on RF classifier
class_weights_plot(RandomForestClassifier, 'Class Weights Influence on Random Forest of X2')

# Grid Search Hyperparameters for Random Forest
rfc = RandomForestClassifier(random_state=1111)

param_grid = {
        "n_estimators" : [5, 6, 7, 8, 9],
        "max_depth" : [7, 8, 9, 10],
        "min_samples_leaf" : [7, 8, 9, 10],
        "class_weight": [None, 'balanced', {0: 1, 1: 5}]}

CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=5, scoring='average_precision')
CV_rfc.fit(X2, y)
print(CV_rfc.best_params_)
print('Best average precision score: {:.4f}'.format(CV_rfc.best_score_))
```