One Year Life Expectancy Post Thoracic Surgery Using Machine Learning

NAME : BANDI.MOHAN SAI
REG.NO : 39110128
Sathyabama Institute of Science and Technology
Computer Science and Engineering

INTRODUCTION

1.1.overview

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

1.2 purpose

the United States, lung cancer claims more lives every year than colon cancer Lung cancer is the leading cause of cancer-related deaths in the world. In, prostate cancer, and breast cancer combined. Despite the very serious prognosis (outlook) of lung cancer, some people with earlier-stage cancers are cured. More than 430,000 people alive today have been diagnosed with lung cancer at some point. The data is dedicated to classification problems related to the post-operative life expectancy in lung cancer patients: class 1 - death within one year after surgery, class 2 - survival. We will be using classification algorithms such as Decision tree, Random forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

2.LITERATURE SURVEY

2.1 Existing problem

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%. 2.1 proposed solution A system is developed which detects the lung cancer from the given input CT scanned lung images which are in DICOM (.dcm) format. In addition to this, the system also helps to predict the Life Expectancy Post Thoracic Surgery of the Lung Cancer infected patients.

3.THEORITICAL ANALYSIS

3.1 Block diagram

3.2 Hardware/software designing

* Software Requirements Processor : intel core i3 * Operating System :windows Speed : 1.19GHz * Coding Langugae:Python3.7 RAM : 8.0GB Memory usage : 7.2kb Keyboard : Standard keyboard Monitor : 15 VGA color

4.EXPERIMENTAL INVESTIGATIONS

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

5. FLOWCHART

6.RESULT

7.ADVANTAGES & DISADVANTAGES

8.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 datase.

9.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.

10.BIBILOGRAPHY

As we need to connect to internet for opening websites and gather information about the skills required for the jobs, we can also use other programming languages like C, C++, Java but its effective to use and also have predefined functions to use in the python language since the syntax for this programming language is simple to use and is more effective comparatively. So we have used python for this project for accessing sites and recommended skills as per the present trending technologies

APPENDIX

Source code

import pandas as pd import numpy as np import seaborn as sns from sklearn.model_selection import train_test_split from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score import matplotlib.pyplot as plt from sklearn.linear_model import LogisticRegression from sklearn.ensemble import RandomForestClassifier, GradientBoostingCl assifier from sklearn.tree import DecisionTreeClassifier from sklearn.neighbors import KNeighborsClassifier from sklearn.metrics import f1_score from sklearn.metrics import classification_report, confusion_matrix import itertools df=pd.read_csv("ThoracicSurgery.csv") df.head() df.columns df.describe() df.shape df.info() df.isnull().sum() live = df[df['Death_1yr'] == 0] death = df[df['Death_1yr'] == 1] cond = ['FVC', 'FEV1', 'Performance', 'Pain', 'Haemoptysis', 'Dyspnoea' , 'Cough', 'Weakness',\ 'Tumor_Size', 'Diabetes_Mellitus', 'MI_6mo', 'PAD', 'Smoking', 'Asthma', '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': 1, 'Death 1 yr Mean': 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(df.Dea th_1yr)*100)) ld d = np.array(d) l = np.array(l) p_diff = (d-1)/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', fontsi ze=18) axes[0].set_xticks(cond) axes[0].set_xticklabels(cond, rotation=90) axes[0].set_ylabel('Percent',

```
fontsize=13) tf_col = ['Pain', 'Haemoptysis', 'Dyspnoea', 'Cough', 'Weakness', 'Diab etes_Mellitus', 'MI_6mo', 'PAD', 'Smoking', 'Asthma'] tf_sum = [df[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', fo ntsize=18) plt.tight_layout() plt.show() 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() def permutation_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 def draw_perm_reps(data_1, data_2, func, size=1): """Generate multiple permutation replicates.""" perm_replicates = np.empty(size)  for i in range(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 def diff_of_means(data_1, data_2): """Difference in means of two arrays.""" diff = np.mean(data_1) - np.mean(data_2) return diff condition = ['FVC', 'FEV1', 'Performance', 'Pain', 'Haemoptysis', 'Dysp noea', 'Cough', 'Weakness',\ 'Tumor_Size', 'Diabetes_Mellitus', 'MI_6mo', 'PAD', 'Smoki ng', '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))) fig, axes = plt.subplots(1,2,figsize=(13,5)) axes[0].plot(df.FVC, df.FEV1, linestyle='none', marker='.') axes[0].set_xlabel('FVC', fontsize=13) axes[0].set_ylabel('FEV1', fontsize=13) axes[0].set_title('FVC vs FEV1', fontsize=16) axes[1].plot(df.Age, df.FEV1, linestyle='none', marker='.', label='FEV1 ') axes[1].plot(df.Age, df.FVC, linestyle='none', marker='.', label='FVC') axes[1].set_xlabel('Age', fontsize=13) axes[1].set_ylabel('FEV1, FVC', fontsize=13) axes[1].legend() axes[1].set_title('Age vs FEV1, FVC', fontsize=16) plt.tight_layout() bnp.corrcoef(df.FVC, df.FEV1)[0,1] np.corrcoef(df.Age, df.FVC)[0,1] np.corrcoef(df.Age, df.FEV1)[0,1] def ecdf(data): """Compute ECDF for a one-dimensional array of measurements.""" n = len(data) x = np.sort(data) y = np.arange(1, n+1) / n return x, y x_fvc, y_fvc = ecdf(df.FVC) x_fev1, y_fev1 = ecdf(df.FEV1) x_age, y_age = ecdf(df.Age) fig, axes = plt.subplots(1,2,figsize=(13,5)) axes[0].plot(x_fvc, y_fvc, marker='.', linestyle='none', label='FVC') axes[0].plot(x_fev1, y_fev1, marker='.', linestyle='none', label='FEV1' ) axes[0].set_xlabel('Numerical Value', fontsize=13) axes[0].set_ylabel('ECDF', fontsize=13) axes[0].legend(loc='upper left') axes[0].set_title('ECDF of FVC & FEV1', fontsize=16) axes[1].plot(x_age, y_age, marker='.', linestyle='none', label='Age') axes[1].set_xlabel('Years Old', fontsize=13) axes[1].set_ylabel('ECDF', fontsize=13) axes[1].legend(loc='upper left')
```

```
axes[1].set_title('ECDF of Age', fontsize=16) plt.tight_layout() df.drop(['FVC'],axis=1, inplace=True) x=df.iloc[:,0:15].values y=df.iloc[:,15:16].values x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random _state=0) print('Shape of x_train {}'.format(x_train.shape)) print('Shape of y_train {}'.format(y_train.shape)) print('Shape of x_test {}'.format(x_test.shape))s print('Shape of y_test {}'.format(y_test.shape)) def decisionTree(x_train, x_test, y_train, y_test): dt=DecisionTreeClassifier() dt.fit(x_train,y_train.ravel()) yPred = dt.predict(x_test) print('***DecisionTreeClassifier***') print('Confusion matrix') print(confusion_matrix(y_test,yPred)) print('Classification report') print(classification_report(y_test,yPred)) def randomForest(x_train, x_test, y_train, y_test): rf = RandomForestClassifier() rf.fit(x_train,y_train.ravel()) yPred = rf.predict(x_test) print('***RandomForestClassifier***') print('Confusion matrix') print(confusion_matrix(y_test,yPred)) print('Classification report') print(classification_report(y_test,yPred)) def KNN(x_train, x_test, y_train, y_test): knn = KNeighborsClassifier() knn.fit(x_train,y_train.ravel()) yPred = knn.predict(x_test) print('***KNeighborsClassifier***') print('Confusion matrix') print(confusion_matrix(y_test,yPred)) print('Classification report') print(classification_report(y_test,yPred)) def xgboost(x_train, x_test, y_train, y_test): xg = GradientBoostingClassifier() xg.fit(x_train,y_train.ravel()) yPred = xg.predict(x_test) print('***GrandientBoostingClassifier***') print('Confusion matrix') print(confusion_matrix(y_test,yPred)) print('Classification report') print(classification_report(y_test,yPred)) def compareModel(x_train, x_test, y_train, y_test): decisionTree(x_train, x_test, y_train, y_test) print('-'*100) randomForest(x_train, x_test, y_train, y_test) print('-'*100) KNN(x_train, x_test, y_train, y_test) print('-'*100) xgboost(x_train, x_test, y_train, y_test) compareModel(x_train, x_test, y_train, y_test) from sklearn.model_selection import cross_val_score rf= RandomForestClassifier() rf.fit(x_train,y_train.ravel()) yPred = rf.predict(x_test) f1_score(yPred,y_test,average='weighted') cv = cross_val_score(rf,x,y.ravel(),cv=5) np.mean(cv) import pickle pickle.dump(rf,open(‘model.pkl’,’wb’))
```

INTRODUCTION

1.1.overview

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 differenttypes of cancer,lung cancer is one of the leading causes of death in both men and women.

According to World Health Organization, it was observedthat in the year 2018 2.09 millioncases 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 latedetection of the lung cancer. Also, the treatmentand the

prognosis depend on the type of the lung cancer, the stage and the patient's performance. Once the lung cancer is detected, possibletreatments include thoracic surgery, chemotherapy and radiotherapy

1.2 purpose

the United States, lung cancer claims more lives every year than colon cancer Lungcancer is the leading cause of cancer-related deaths in the world. In, prostate cancer,and breast cancer combined.

Despite the very serious prognosis(outlook) of lung cancer, some people with earlier-stage cancers are cured. More than 430,000 people alive today have been diagnosed with lung cancer at some point. The data is dedicated to classification problemsrelated to the post-operative life expectancy in lung cancer patients: class1 - death within one year after surgery,class 2 - survival.

We will be using classification algorithms such as Decision tree, Random forest,KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration andIBM deployment.

1. LITERATURE SURVEY

a. Existing problem

Disha Sharma et al. (2011), proposed an approach for the early detection of lung cancer by analyzinglungs CT images using Image Processing techniques[1]. The authors used bit-plane slicing, erosion and Weiner filter image processingtechniquesto extract the lung regions from the CT image. Furtherthe extracted lung regions were segmented using Region growing segmentation algorithmand later Rule basedmodel was used to detect the cancerousnodules. With the help of diagnostics indicator, it was observed that the proposed method achieved an overall accuracyof 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 contrastenhancement and linear filtering. Next, the filteredimage was segmentedusing Region growing Segmentation process.Further the featureslike area and color was

given as input to the Fuzzy system which employedfuzzy membership

functionto find the abnormalities. SindhuV et al.(2014), proposed an approach where the authorsaimed to classify the survival of lung cancer patientspost thoracic surgery[3]. The authors used Naïve Bayes, PART, J48, OneR, Random Forest and Decision stump algorithm techniques to classify the target function. The performancemeasurement shows that Random Forestgave high accuracyof 95.65% comparedto other ML techniques used. Prashant Nareshet al.(2014), proposed a methodologyto detect the lung cancer using Image processing and Neural Techniques[4]. Initially the CT image of lung was filtered to remove Gaussianwhite noise and Otsu's thresholdtechnique was used to do the segmentation of the image. The structuraland 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%. KwetisheJoro Danjuma (2015), proposed a methodology to predict the one-year survivalof the patient post thoracicsurgery[5]. Naïve Bayes, J48 and Multilayer Perceptronalgorithms were used to classifythe target class. The Naïve Bayes gave an accuracyof 74.4%, J48 gave an accuracy of 81.8% and MLP gave an accuracy of 82.4%.

2.1 proposed solution

A system is developed which detects the lung cancer from the given input CT scannedlung images which are in DICOM (.dcm) format. In addition to this, the system also helps to predict the Life ExpectancyPost Thoracic Surgery of the LungCancerinfected patients.

2. THEORITICAL ANALYSIS

a. **Block diagram**

b. Hardware/software designing * Software Requirements

Processor : intel core i3 * OperatingSystem :windows

Speed : 1.19GHz *

CodingLangugae:Python3.7RAM : 8.0GB

Memory usage : 7.2kb

Keyboard :

StandardkeyboardMonitor

: 15 VGA color

3. EXPERIMENTAL INVESTIGATIONS

This is the second part of the system which aims at predicting the survival of lung cancer infected patientpost thoracic surgery.The dataset includes17 attributes whichare specified in Table-1. Amongall those attributes, Risk1Y is the target class specifying zero if the patientsurvives 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 basedon the Information Gain (IG) attribute evaluation which is used to find the importance of an attributeby using the Information Gain with respectto the target class. IG (Class, Attribute) =E (Class)– E (Class | Attribute) where, E standsfor Entropy After the IG Attribute Evaluationon all the 16 independent attributes in the dataset, it is found that the attributes PRE19 andPRE32 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 totrain the model

4. FLOWCHART

5. **RESULT**

6. ADVANTAGES & DISADVANTAGES

7. CONCLUSION

Detection of lung cancer is one of the challenging problems in medicalfield due tostructure 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 containstwo parts.One is Lung Cancer Detectionpart and the other is the Prediction of Life ExpectancyPost Thoracic Surgery. Both the parts can run independently. The systemis 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 infectedwith Lung Cancer. At the backend the image uploaded is preprocessed, segmented and predictedusingthe CNN model which is already trained. Also, the system provides a form requestingfor the 14 attributes requiredto 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 highestaccuracy 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 (MagneticResonance Imaging) or PET (PositionEmission Tomography) as input.
Also, in the prediction of survival part, higher accuracies of the classifiercan beachieved by considering large amount of datase.

8. FUTURE SCOPE

Thoracic surgery is living an enthusiastic era of thriving innovations. The changes inthe fields of diagnostics, in the therapeutic options, the development of surgical techniques and the adoptionof novel technologies has radicallychanged the horizonsof our specialty. We must embrace the innovations, learn navigational bronchoscopy, understand the multipletargeted therapies, and learn new ways to operate on advanced cases. We will not be able to stay complacent with our current3 ports video-assisted thoracoscopic surgery(VATS) technique with open instruments, therefore every surgeon needs to be watchingfor each latest development as it happens.In this brief manuscript we will summarizesome of the most importantdevelopment over the recent years and forecast how this will impacton the patients affected by thoracic malignancies. As thoracic surgeonswe have to embraces these developments in order to redefine our specialty.

9. BIBILOGRAPHY

As we need to connectto internet for opening websitesand gather information aboutthe skills required for the jobs, we can also use other programming languages like C,C++, Java but its effective to use and also have predefined functionsto use in the python language since the syntax for this programming language is simple to use and is more effectivecomparatively. So we have used python for this projectfor accessing sites and recommended skills as per the present trending technologies

APPENDIX

A.Source code

```
import pandas
as pd import
numpy as np
importseaborn
as sns
from sklearn.model_selection import
train_test_splitfrom sklearn.linear_model
import LogisticRegressionfrom sklearn.metrics
importaccuracy_score
import matplotlib.pyplot as plt
from sklearn.linear_model importLogisticRegression
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.tree import
DecisionTreeClassifier from
```

```
sklearn.neighbors import
KNeighborsClassifierfrom sklearn.metrics
importf1_score
from sklearn.metrics import classification_report,
confusion_matriximport itertools
df=pd.read_csv("ThoracicSurger
y.csv") df.head()
df.columns
df.describ
e()
df.shape
df.info()
df.isnull(
).sum()
live  =
df[df['Death_1yr'] == 0]
death= df[df['Death_1yr']
== 1]

cond = ['FVC','FEV1', 'Performance', 'Pain','Haemoptysis', 'Dyspnoea'
, 'Cough', 'Weakness',\


        'Tumor_Size', 'Diabetes_Mellitus', 'MI_6mo', 'PAD',
'Smoking','Asthma', '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': 1,
'Death 1yr Mean':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(df.Death_1yr)*100))
ld
d =
np.array
(d)l =
np.array
(l)

p_diff = (d-1)/l*100
fig, axes =

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

axes[0].bar(cond, p_diff)
```

```
axes[0].set_title('Mean Difference % betweenDead and Live 1yr', fontsi
ze=18)
axes[0].set_xticks(c
ond)
axes[0].set_xticklabels(cond,
rotation=90)
axes[0].set_ylabel('Percent',
fontsize=13)

tf_col = ['Pain', 'Haemoptysis', 'Dyspnoea', 'Cough', 'Weakness',
'Diabetes_Mellitus', 'MI_6mo','PAD', 'Smoking', 'Asthma']
tf_sum = [df[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_layo

ut()plt.show()

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()
def permutation_sample(data1, data2):
    """Generate a permutation samplefrom 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

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

```
    for i in range(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

def diff_of_means(data_1, data_2):
    """Difference in means of two
    arrays."""diff= np.mean(data_1) -
    np.mean(data_2)returndiff
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)))
fig, axes = plt.subplots(1,2,figsize=(13,5))
axes[0].plot(df.FVC, df.FEV1,linestyle='none',
marker='.')

axes[0].set_xlabel('FVC', fontsize=13)
axes[0].set_ylabel('FEV1',
fontsize=13) axes[0].set_title('FVC vs
FEV1', fontsize=16)

axes[1].plot(df.Age, df.FEV1, linestyle='none', marker='.',
label='FEV1')
axes[1].plot(df.Age, df.FVC, linestyle='none', marker='.',
label='FVC')axes[1].set_xlabel('Age', fontsize=13)
axes[1].set_ylabel('FEV1, FVC',
fontsize=13)axes[1].legend()
axes[1].set_title('Age vs FEV1, FVC',fontsize=16)

plt.tight_layout()
```

```
bnp.corrcoef(df.FVC,
df.FEV1)[0,1]
np.corrcoef(df.Age,
df.FVC)[0,1]


np.corrcoef(df.Age,
df.FEV1)[0,1]def
ecdf(data):
    """Compute ECDF for a one-dimensional array of
    measurements."""n = len(data)
    x = np.sort(data)
    y = np.arange(1,
    n+1) / nreturn x,
    y
x_fvc, y_fvc =
ecdf(df.FVC) x_fev1,
y_fev1= ecdf(df.FEV1)
x_age,y_age =
ecdf(df.Age)

fig, axes = plt.subplots(1,2,figsize=(13,5))
axes[0].plot(x_fvc, y_fvc, marker='.', linestyle='none',
label='FVC') axes[0].plot(x_fev1, y_fev1,marker='.',
linestyle='none', label='FEV1'
)

axes[0].set_xlabel('Numerical Value',
fontsize=13) axes[0].set_ylabel('ECDF',
fontsize=13) axes[0].legend(loc='upper left')
axes[0].set_title('ECDF of FVC & FEV1',
fontsize=16)

axes[1].plot(x_age, y_age, marker='.', linestyle='none',
label='Age')axes[1].set_xlabel('Years Old', fontsize=13)
axes[1].set_ylabel('ECDF', fontsize=13)
axes[1].legend(loc='upper left')
axes[1].set_title('ECDF of Age',
fontsize=16)

plt.tight_layout()
df.drop(['FVC'],axis=1,
inplace=True)
x=df.iloc[:,0:15].values
y=df.iloc[:,15:16].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random
_state=0)
print('Shape of x_train
{}'.format(x_train.shape))print('Shape of
y_train {}'.format(y_train.shape))
print('Shape of x_test
{}'.format(x_test.shape))s print('Shape of
```

```
y_test {}'.format(y_test.shape)) def
decisionTree(x_train, x_test,y_train,
y_test):
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train.ravel())
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifie
    r***') print('Confusion
    matrix')
    print(confusion_matrix(y_test,
    yPred)) print('Classification
    report')
    print(classification_report(y_test,yPred))
def randomForest(x_train, x_test,y_train,
    y_test):rf = RandomForestClassifier()
    rf.fit(x_train,y_train.ravel())
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
def KNN(x_train, x_test,y_train,
    y_test):knn =
    KNeighborsClassifier()
    knn.fit(x_train,y_train.ravel())


    yPred = knn.predict(x_test)
    print('***KNeighborsClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
def xgboost(x_train, x_test,y_train,
    y_test):xg =
    GradientBoostingClassifier()
    xg.fit(x_train,y_train.ravel())
    yPred = xg.predict(x_test)
    print('***GrandientBoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
def compareModel(x_train, x_test,y_train,
    y_test):decisionTree(x_train, x_test,
    y_train, y_test)print('-'*100)
    randomForest(x_train, x_test,
    y_train,y_test)print('-'*100)
    KNN(x_train, x_test,y_train,
    y_test)print('-'*100)
    xgboost(x_train, x_test, y_train,
y_test) compareModel(x_train, x_test,
```

```
y_train, y_test)from sklearn.model_selection
importcross_val_scorerf=
RandomForestClassifier()
rf.fit(x_train,y_train.ravel())
yPred = rf.predict(x_test)
f1_score(yPred,y_test,average='wei
ghted') cv =
cross_val_score(rf,x,y.ravel(),cv=
5)np.mean(cv)
import pickle
pickle.dump(rf,open(‘model.pkl
’,’wb’))
```