IT4060 - MACHINE LEARNING

Assignment 2

Lung Cancer classifier using Random Forest Classifier algorithm

B.Sc. (Hons) Degree in Information Technology Specializing in

Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology Sri Lanka

Group Details: -

ID number	Name
IT18160512	Ganegoda R.D.
IT19234694	S.D. Thewahettige
IT19214962	Alagiyawanna A.M.A.R.P.
IT19233840	Pethiyagoda R.M.S.U.B.

Contents

Table of Figures

1. Introduction

Lungs are the significant organs of the respiratory system of Animals. So, the lungs assist to provide continues oxygen supply for every living animal. Lung cancers are basically beginning in the lungs and spreading to the body. Lungs are very spongy organ. And it helps to inhale oxygen from outside and release carbon dioxide when exhale. This is a dangerous cancer in such cases it can leading cause of cancer deaths worldwide. There are several facts that can cause to the lung cancer. Those can be display as below,

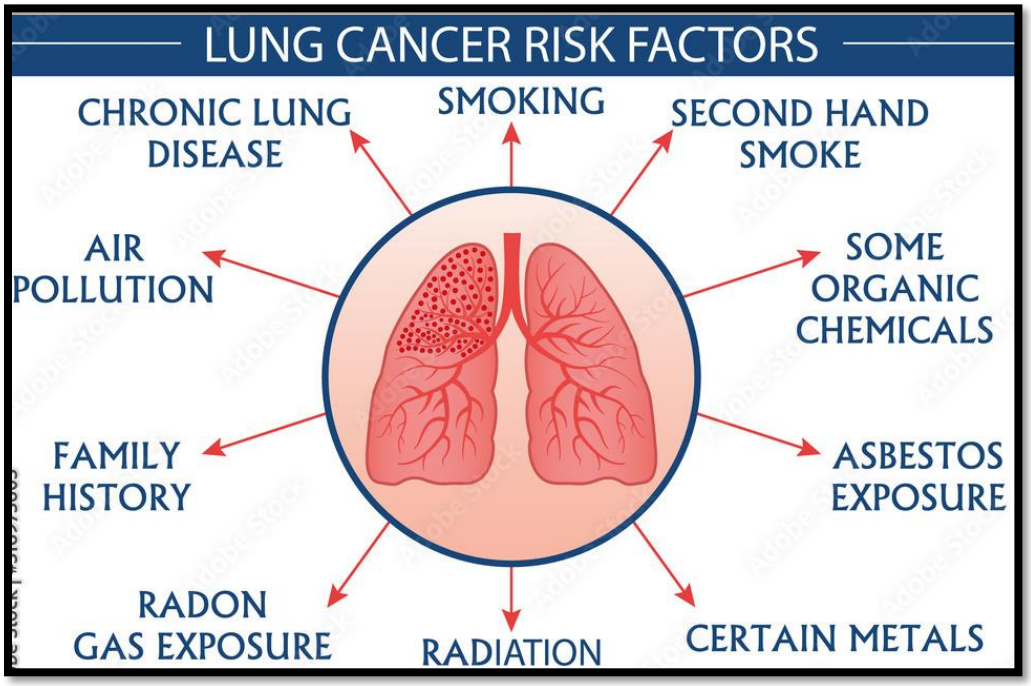


Figure 1.1: Facts cause for the Lung Cancer

The lung cancer spreading and appointed to the risky situation is a multi-stage and multi-step process.

Lung Nodules

Lung nodules are small masses of tissue. They can be benign, precancerous or metastatic tumors. Lung nodules are often found when a patient is being tested for unrelated symptoms, such as abdominal pain or an injury.

Non-Small Cell Lung Cancer

This cancer is most common cancer among other lung cancers. It grows and spreads more slowly than small cell lung cancer. There are 3 types of non-smaller cell lung cancers they are Adenocarcinoma, Large cell carcinomas and Squamous cell carcinoma. Adenocarcinoma is usually beginning along the outer sections of the lungs and it also infected to the non-smokers. Large cell carcinomas may begin anywhere in the lungs and tend to grow quickly. Squamous cell carcinoma often begins in the bronchi near the middle of the lungs.

Small Cell Lung Cancer

Almost all cases of small cell lung cancer are due to cigarette smoking. It is fast-growing cancer when comparing to other lung cancers. There are two types of small cell lung cancers. They are Small cell carcinoma and Combined small cell carcinoma. Combined small cell carcinoma tends to spread more quickly to other parts of the body.

Mesothelioma

This is a rare cancer of the chest lining, most often caused by asbestos exposure. It accounts for about 5 percent of all lung cancer cases. Mesothelioma develops over a long period time, from 30 to 50 years between exposure to asbestos.

The lung cancer diagnosis can be classified as below. There are two factors,

Benign(B) – This is not cancer they belong to healthy appearance Benign tumors may grow larger but do not spread to other parts of the body.

Malignant(M) – This is a cancer. Malignant cells grow in an uncontrolled way and can spread to other parts of the body.

Early lung cancers detection system is a way to deal with controlling in cancers in primary stage. Nowadays early detection and reduction of mortality rates are handled in different kind of ways. We are trying to develop a system for early detection of patients by assessing several factors affecting the lung cancers using Machine learning approaches.

2. Dataset

In order to develop an early detection system for lung cancer, the data set was taken from the Kaggle website [1]

DATASET: https://www.kaggle.com/datasets/prashanthpacchi/classification-datasetcancer-prediction

Attribute Characteristics	Real
Number of Instances	569
Number of Attributes	30
Missing Values	No
Associated Tasks	Classification
Class distribution	Benign (357) , Malignant (212)

Figure 2.1:Attribute Characters according to the dataset

Real-valued features	Description
Radius	mean of distances from center to points on the perimeter
Texture	standard deviation of gray-scale values
Perimeter	Boundary around a shape
Area	measurement of a surface
Smoothness	local variation in radius lengths
Compactness	perimeter^2 / area - 1.0
Concavity	severity of concave portions of the contour
Concave points	number of concave portions of the contour
Symmetry	Correct correspondence between different things
Fractal dimension	"coastline approximation" - 1

Figure 2.2:Real Valued features according to the dataset

3. Methodology

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.[2]

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees.[3] Increasing the number of trees increases the precision of the outcome.

A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision.[4] It generates predictions without requiring many configurations in packages

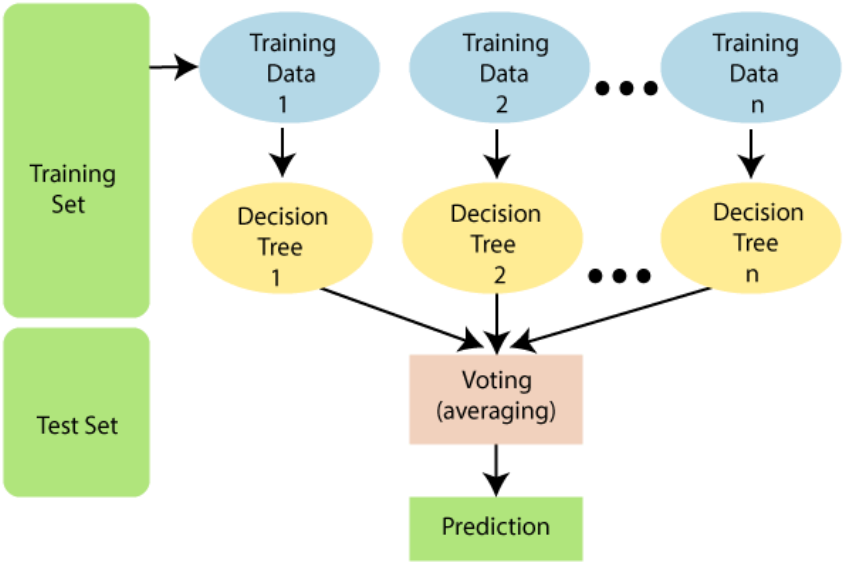


Figure 3.1:Workflow of the random-forest classification

4. Implementations

4.1 Importing data

As the first step we have to import the required python libraries. Commonly we are importing Pandas, Numpy, Matplotlib, Seaborn and Sklearn libraries. Pandas library is used to analyze data and machine learning related tasks. Numpy perform a wide variety of mathematical operations on arrays. Matplotlib library is used to graphical plotting and data visualization. Seaborn is used to exploratory data analysis and data visualization purposes. It can easily work with data frames. Sklearn provides an efficient tool for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. Imported common libraries can be shown as 4.1 figure.

```
In [1]:
#Import all libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import sklearn.metrics as sm
%matplotlib inline
```

Figure 4.1:Import required libraries

Then we can load the data from csv file using below command according to the 4.2 figure. Data frame is named as **lung_cancer** and then use **read_csv** function to load the dataset.

```
In [2]:
# load the dataset from csv file
lung_cancer = pd.read_csv("LungCancerData.csv")
```

Figure 4.2:Load the data from csv

Then to read first five rows of the data set we can use the below command as 4.3 figure.

```
In [3]:
#Display first 5 rows of the dataset
lung_cancer.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	...	rad
0	87139402	B	12.32	12.39	78.85	464.1	0.10280	0.06981	0.03987	0.03700	...	
1	8910251	B	10.60	18.95	69.28	346.4	0.09688	0.11470	0.06387	0.02642	...	
2	905520	B	11.04	16.83	70.92	373.2	0.10770	0.07804	0.03046	0.02480	...	
3	868871	B	11.28	13.39	73.00	384.8	0.11640	0.11360	0.04635	0.04796	...	
4	9012568	B	15.19	13.21	97.65	711.8	0.07963	0.06934	0.03393	0.02657	...	

5 rows × 32 columns

Figure 4.3:Read first five rows of the data set

4.2 Exploring data

In this exploring step it will discover the structure and the content according to the dataset. In that case we can use several functions in pandas library. such as shape(), dtypes (), keys(), describe() etc. Then to view the all column names of the data set we use keys() function as 4.5 figure.

```
In [4]:
# Key values in the dataset
lung_cancer.keys()

Out[4]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'points_mean', 'symmetry_mean', 'dimension_mean', 'radius_se',
       'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'points_se', 'symmetry_se',
       'dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst',
       'area_worst', 'smoothness_worst', 'compactness_worst',
       'concavity_worst', 'points_worst', 'symmetry_worst', 'dimension_worst'],
      dtype='object')
```

Figure 4.4:key columns of the dataset

Then to get the counts of Benign(B) and Malignant(M) values according to the diagnosis column can use the below command as 4.5 figure.

```
In [5]:
#get the counts of Benign and Malignant values in the diagnosis column
lung_cancer.diagnosis.value_counts()

Out[5]: B    357
        M    212
        Name: diagnosis, dtype: int64
```

Figure 4.5: diagnosis counts according to classification

To view the shape of the dataset can use shape function as below 4.6 figure.

```
In [6]:
# Get the shape according to the columns and rows counts
lung_cancer.shape

Out[6]: (569, 32)
```

Figure 4.6: shape according to the rows and columns

To view the data types of each column can use the command as the 4.7 figure.

```
In [7]:
# Display the data types
lung_cancer.dtypes

Out[7]:
id                       int64
diagnosis               object
radius_mean            float64
texture_mean           float64
perimeter_mean         float64
area_mean              float64
smoothness_mean        float64
compactness_mean       float64
concavity_mean         float64
points_mean            float64
symmetry_mean          float64
dimension_mean         float64
radius_se              float64
texture_se             float64
perimeter_se           float64
area_se                float64
smoothness_se          float64
compactness_se         float64
concavity_se           float64
points_se              float64
symmetry_se            float64
dimension_se           float64
radius_worst           float64
texture_worst          float64
perimeter_worst        float64
area_worst             float64
smoothness_worst       float64
compactness_worst      float64
concavity_worst        float64
points_worst           float64
symmetry_worst         float64
dimension_worst        float64
dtype: object
```

Figure 4.7:Data types

Then to get a descriptive statistic for each column can be code as 4.8 figure and the output also display as below.

```
In [8]:
# Descriptive statistics for each column
lung_cancer.describe()

Out[8]:
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	symmetr
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	

8 rows × 31 columns

Figure 4.8: Describe statistics of each column

4.3 Data preprocessing

We can get the null value count according to the dataset and by using below command it will give the count of null values in the dataset columns.

```
In [9]:
# Count the empty values in each column
lung_cancer.isnull().sum()

Out[9]:
id                    0
diagnosis             0
radius_mean           0
texture_mean          0
perimeter_mean        0
area_mean             0
smoothness_mean       0
compactness_mean      0
concavity_mean        0
points_mean           0
symmetry_mean         0
dimension_mean        0
radius_se             0
texture_se            0
perimeter_se          0
area_se               0
smoothness_se         0
compactness_se        0
concavity_se          0
points_se             0
symmetry_se           0
dimension_se          0
radius_worst          0
texture_worst         0
perimeter_worst       0
area_worst            0
smoothness_worst      0
compactness_worst     0
concavity_worst       0
points_worst          0
symmetry_worst        0
dimension_worst       0
dtype: int64
```

Figure 4.9:Null values of the dataset

Checking if any duplicates in "id" column by finding unique values and the frequency is greater than 1. ID column can drop from the table because it is not helping to create lung cancer predictions. It can drop as below. So, it helps to clean the dataset.

```
In [21]:
# Checking if any duplicates in "id" column by finding unique values and if their frequency is greater than 1
lung_cancer.id.value_counts().unique()

Out[21]: array([1], dtype=int64)

In [22]:
# Since "id" column has no direct impact on target value so it will drop from dataset
lung_cancer.drop('id',axis=1,inplace=True)
```

Figure 4.10: Drop ID column and checking duplicate IDs

Then check the duplicate values as below.

```
In [24]:
lung_cancer.duplicated().sum()

Out[24]: 0

In [25]:
# Checking if any duplicate values in the df
lung_cancer.duplicated()

Out[25]: 0      False
         1      False
         2      False
         3      False
         4      False
                ...
         564    False
         565    False
         566    False
         567    False
         568    False
         Length: 569, dtype: bool
```

Figure 4.11:Check duplicate values

The cleaned dataset can be displayed as below it shows by removing ID column.

```
In [28]:
lung_cancer.shape

Out[28]: (569, 31)
```

Figure 4.12:Shape of the data set

According to the figure 4.7 it says that “diagnosis” column data type is represent as object data type. So, we have to transform the data type. So, all objective data type is encrypted in “diagnosis” column values as ‘M’ to 1 and ‘B’ to 0. It can be displayed as below.

```
In [29]:
# Mapping string values of the diagnosis column feature to numerical value feature using map function
cancer_mapping = {'B':0, 'M':1}
lung_cancer.diagnosis = lung_cancer.diagnosis.map(cancer_mapping)
```

Figure 4.13:mapping string values to numeric values

Displaying the data set after cleaning. It will show as below.

```
In [30]:
# afer maping show first 20 raws of the data frame
lung_cancer.head(20)
```

Out[30]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	symmetry_mean
0	0	12.32	12.39	78.85	464.1	0.10280	0.06981	0.039870	0.037000	0.1959
1	0	10.60	18.95	69.28	346.4	0.09688	0.11470	0.063870	0.026420	0.1922
2	0	11.04	16.83	70.92	373.2	0.10770	0.07804	0.030460	0.024800	0.1714
3	0	11.28	13.39	73.00	384.8	0.11640	0.11360	0.046350	0.047960	0.1771
4	0	15.19	13.21	97.65	711.8	0.07963	0.06934	0.033930	0.026570	0.1721
5	0	11.57	19.04	74.20	409.7	0.08546	0.07722	0.054850	0.014280	0.2031
6	0	11.51	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0.1388
7	1	13.81	23.75	91.56	597.8	0.13230	0.17680	0.155800	0.091760	0.2251
8	0	10.49	19.29	67.41	336.1	0.09989	0.08578	0.029950	0.012010	0.2217
9	0	11.06	14.96	71.49	373.9	0.10330	0.09097	0.053970	0.033410	0.1776
10	1	20.59	21.24	137.80	1320.0	0.10850	0.16440	0.218800	0.112100	0.1848
11	0	12.25	17.94	78.27	460.3	0.08654	0.06679	0.038850	0.023310	0.1970
12	0	13.14	20.74	85.98	536.9	0.08675	0.10890	0.108500	0.035100	0.1562
13	0	13.05	19.31	82.61	527.2	0.08060	0.03789	0.000692	0.004167	0.1819
14	1	19.59	25.00	127.70	1191.0	0.10320	0.09871	0.165500	0.090630	0.1663
15	0	14.59	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.037360	0.1454
16	0	15.71	13.93	102.00	761.7	0.09462	0.09462	0.071350	0.059330	0.1816
17	0	12.67	17.30	81.25	489.9	0.10280	0.07664	0.031930	0.021070	0.1707
18	1	20.09	23.86	134.70	1247.0	0.10800	0.18380	0.228300	0.128000	0.2249
19	0	12.19	13.29	79.08	455.8	0.10660	0.09509	0.028550	0.028820	0.1880

20 rows × 31 columns

Figure 4.14: Displaying first 20 rows

Visualizing selected features by target can be displayed as below figure it shows the visualization according to several features of the dataset.

```
In [21]:
# Visualising selected features by target:
sb.pairplot(lung_cancer, hue = 'diagnosis', vars = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothnes
```

```
Out[21]: <seaborn.axisgrid.PairGrid at 0x15dc1bf8940>
```

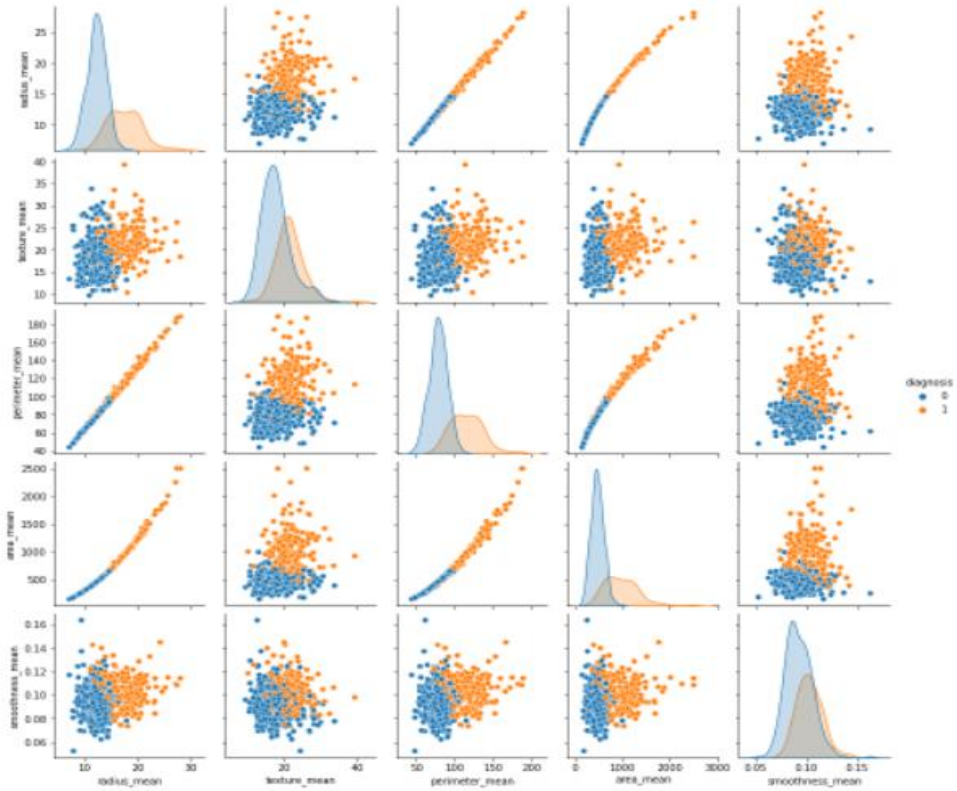


Figure 4.15: Visualizing selected features by target

Getting insight of data distribution based on frequency of unique values in the features can display using below figure.

```
In [22]:
# Getting insight of data distribution based on frequency of unique values in the features
lung_cancer.hist(figsize=(20,20))
plt.show()
```

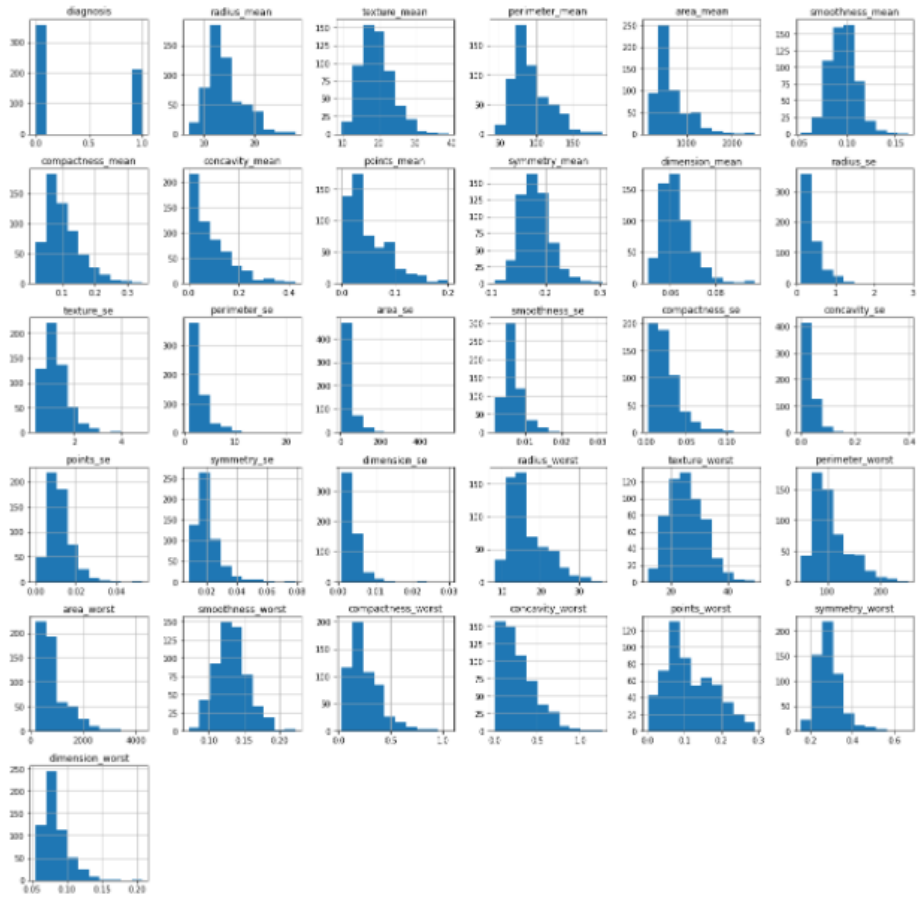


Figure 4.16: Insight frequency of unique values in the features

4.4 Split the data

In the data set, it can be divided into two sections They are feature and target. Feature will get as the independent (X) variable and the target will identify as the dependent (Y) variable of the dataset. According to the data set diagnosis column is considered as target data, the dependent variable. The other columns are defined as feature section. Then splitting the dataset to ‘x’ and ‘y’ again as training and testing data.

```
In [36]:
# drop the target label coloumns
X = lung_cancer.drop(['diagnosis'],axis=1)
```

```
In [37]:
X
```

Out[37]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	symmetry_mean	dimensic
0	12.32	12.39	78.85	464.1	0.10280	0.06981	0.03987	0.03700	0.1959	
1	10.60	18.95	69.28	346.4	0.09688	0.11470	0.06387	0.02642	0.1922	
2	11.04	16.83	70.92	373.2	0.10770	0.07804	0.03046	0.02480	0.1714	
3	11.28	13.39	73.00	384.8	0.11640	0.11360	0.04635	0.04796	0.1771	
4	15.19	13.21	97.65	711.8	0.07963	0.06934	0.03393	0.02657	0.1721	
...	...	...	...	...	...	...	...	...	...	
564	13.17	18.22	84.28	537.3	0.07466	0.05994	0.04859	0.02870	0.1454	
565	10.26	14.71	66.20	321.6	0.09882	0.09159	0.03581	0.02037	0.1633	
566	15.28	22.41	98.92	710.6	0.09057	0.10520	0.05375	0.03263	0.1727	
567	14.53	13.98	93.86	644.2	0.10990	0.09242	0.06895	0.06495	0.1650	
568	21.37	15.10	141.30	1386.0	0.10010	0.15150	0.19320	0.12550	0.1973	

569 rows × 30 columns

Figure 4.17: Drop the target labeled column

```
In [38]:
# Define output values - this is the target
Y = lung_cancer['diagnosis']
Y
```

```
Out[38]: 0      0
         1      0
         2      0
         3      0
         4      0
               ..
         564    0
         565    0
         566    1
         567    0
         568    1
         Name: diagnosis, Length: 569, dtype: int64
```

Figure 4.18: Define output values of the target

```
In [41]:
# splitting the data into test and train / using 20% of the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state=5)
```

Figure 4.19:Spliting data to training and testing data set

According to the 4.15 figure it shows that the training data set and testing dataset was divided using the data array. The dividing function was called from the Sklearn library and the used function was train_test_split function. In that function there are several parameters they are ‘X’ and ‘Y’ parameters, test_size specify the size of the testing dataset. According to the above figure it displayed as 20% and the training data set has 80%. Random state is defined as 5. And it will be divided it into 5 random subsets.

In [42]:
```
X_train
```

Out[42]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	symmetry_mean	dimensic
306	15.220	30.62	103.40	716.9	0.10480	0.20870	0.25500	0.09429	0.2128	
410	12.890	14.11	84.95	512.2	0.08760	0.13460	0.13740	0.03980	0.1596	
197	14.620	24.02	94.57	662.7	0.08974	0.08606	0.03102	0.02957	0.1685	
376	14.740	25.42	94.70	668.6	0.08275	0.07214	0.04105	0.03027	0.1840	
244	19.730	19.82	130.70	1206.0	0.10620	0.18490	0.24170	0.09740	0.1733	
...	...	...	...	...	...	...	...	...	...	
8	10.490	19.29	67.41	336.1	0.09989	0.08578	0.02995	0.01201	0.2217	
73	17.850	13.23	114.60	992.1	0.07838	0.06217	0.04445	0.04178	0.1220	
400	9.847	15.68	63.00	293.2	0.09492	0.08419	0.02330	0.02416	0.1387	
118	19.800	21.56	129.70	1230.0	0.09383	0.13060	0.12720	0.08691	0.2094	
206	11.600	18.36	73.88	412.7	0.08508	0.05855	0.03367	0.01777	0.1516	

455 rows × 30 columns

In [43]:
```
X_test
```

Out[43]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	symmetry_mean	dimensic
28	9.606	16.84	61.64	280.5	0.08481	0.09228	0.084220	0.022920	0.2036	
163	9.000	14.40	56.36	246.3	0.07005	0.03116	0.003681	0.003472	0.1788	
123	21.710	17.25	140.90	1546.0	0.09384	0.08562	0.116800	0.084650	0.1717	
361	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0.1847	
549	12.050	14.63	78.04	449.3	0.10310	0.09092	0.065920	0.027490	0.1675	
...	...	...	...	...	...	...	...	...	...	
414	11.740	14.02	74.24	427.3	0.07813	0.04340	0.022450	0.027630	0.2101	
515	12.830	22.33	85.26	503.2	0.10880	0.17990	0.169500	0.068610	0.2123	
186	17.050	19.08	113.40	895.0	0.11410	0.15720	0.191000	0.109000	0.2131	
3	11.280	13.39	73.00	384.8	0.11640	0.11360	0.046350	0.047960	0.1771	
261	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059990	0.027380	0.1593	

114 rows × 30 columns

Figure 4.20: X_train and X_test data

In [44]:
```
Y_train
```

Out[44]:
```
306    1
410    0
197    0
376    0
244    1
      ..
8      0
73     0
400    0
118    1
206    0
Name: diagnosis, Length: 455, dtype: int64
```

In [45]:
```
Y_test
```

Out[45]:
```
28     0
163    0
123    1
361    1
549    0
      ..
414    0
515    1
186    1
3      0
261    0
Name: diagnosis, Length: 114, dtype: int64
```

Figure 4.21:Y_train and Y_test data

4.5 Feature Scaling

To get the same magnitude point we have to scale the data. This describe that the feature variable in several ranges. So, the data scaling can be shown as below.

```
In [46]:
print(X_train.shape,' The shape of Training Features')
print(Y_train.shape,' The shape of Training Lables')
print(X_test.shape,'  The shape of Testing Features')
print(Y_test.shape,'  The shape of Testing Lables')

(455, 30)  The shape of Training Features
(455,)  The shape of Training Lables
(114, 30)   The shape of Testing Features
(114,)   The shape of Testing Lables
```

Figure 4.22:Scaling the feature variable

4.6 Build the Random Forest classifier

The random forest is a classification algorithm consisting of many decisions trees. And also, it is versatile classification tool that makes an aggregated prediction using a group of decision trees trained using the bootstrap method with extra randomness while growing trees by searching for the best features among a randomly selected feature subset.

According to the below figures to create classifications we have used the RandomForestClassifier() function. It has some parameters. They are n_estimators, criteria, support criteria and random_state. n_estimators parameter is used to process the number of trees in the forest. Then the criteria is used to measure the consistency of each divisions. Random_state is used to control the randomness.

There are so many approaches that we can use to train the models to classify. According to the above 4.13 figure it shows the accuracy score as 0.99780 for training dataset. This model is used to predict the cancer from patients.

```
In [38]:
# Using Random Forest Classification algorithm
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state =
0)
rf_classifier.fit(X_train, Y_train)

Out[38]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

In [39]:
# Print model accuracy on the training data
print('Random Forest Classifier Training Accuracy:', rf_classifier.score(X_train, Y_train))

Random Forest Classifier Training Accuracy: 0.9978021978021978
```

Figure 4.23: Train the random forest classifier model

4.7 Testing

To create the testing data set we have used 20% from all data. According to the below figure it shows that the testing accuracy score is 0.947368.

```
In [41]:
# Get the accuracy of the test data
Y_prediction = rf_classifier.predict(X_test)
test_accuracy=sm.accuracy_score(Y_test, Y_prediction)
print('Testing Accuracy:',str(test_accuracy))

Testing Accuracy: 0.9473684210526315
```

Figure 4.24:Testing accuracy score

4.8 Classification report

This classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model. This report provides a depth understanding about the global accuracy. This can conceal the functional limitations in a multi-level problem class. According to the below figure it shows that the cancer dataset of the report. The classifier report can be shown as below,

```
In [43]:
#Generate classification report based on the predicted values from dataset
from sklearn import metrics
print("Classification Report : \n\n", metrics.classification_report(Y_prediction, Y_test,
target_names = ["Malignant","Benign"]))

Classification Report :

               precision    recall  f1-score   support

   Malignant       0.97      0.95      0.96        74
      Benign       0.90      0.95      0.93        40

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.94       114
weighted avg       0.95      0.95      0.95       114
```

Figure 4.25:Classification Report

Precision - Ratio of true positives to the sum of true and false positives.

Recall - Ratio of true positives to the sum of true positives and false negatives.

F1-score - Gives the class accuracy related to other classes.

Support - Enter the actual number of class occurrences in the specified dataset. It just diagnoses the performance evaluation process.

4.9 Visualization

Get a count of patients with cancer infected Malignant (M) and not cancer infected Benign (B) cells by the dataset and Visualize using a count plot according to the figure 4.26 below.

```
In [23]:
# 357 benign (1), 212 malignant (0)
sb.countplot(lung_cancer['diagnosis'], label = "Count")
```

```
C:\Users\ACER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[23]: <AxesSubplot:xlabel='diagnosis', ylabel='count'>
```

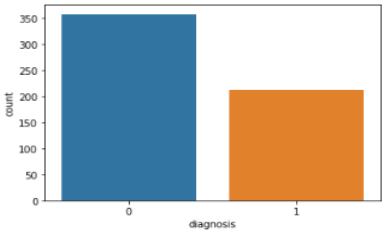


Figure 4.26: Visualizing the count plot

```
In [24]:
sb.scatterplot(x = 'area_mean', y = 'smoothness_mean', hue = 'diagnosis', data = lung_cancer)
```

```
Out[24]: <AxesSubplot:xlabel='area_mean', ylabel='smoothness_mean'>
```

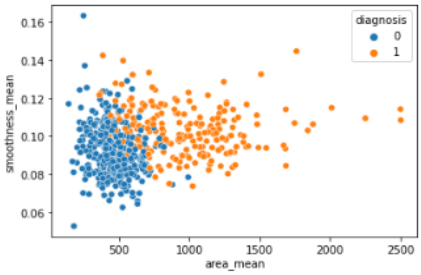


Figure 4.27: scatterplot of the dataset

Get the correlation of the columns in lung cancer dataset and Visualizing the correlation by creating a heat map as below.

```
In [25]:
# Visualize the correlation
plt.figure(figsize=(24,13))
sb.heatmap(lung_cancer.corr(), annot=True)
```

```
Out[25]: <AxesSubplot:>
```

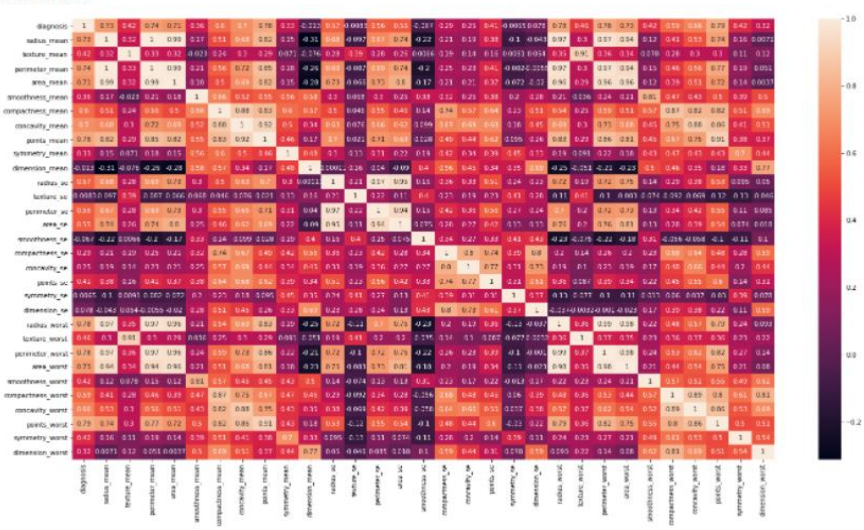


Figure 4.28:Visualize the correlation

According to the below figure 4.29 despites the confusion matrix for the test results. The confusion matrix shows the number of correct disease diagnosis data and not correct disease diagnosis data, the true positive and the true negative value as below.

```
In [44]:
# Based on the test values this code phase generate the confusion matrix
from sklearn.metrics import confusion_matrix
sb.set()
get_ipython().run_line_magic('matplotlib', 'inline')
matrx = confusion_matrix(Y_test, Y_prediction)
sb.heatmap(matrx.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true class')
plt.ylabel('predicted class')

Out[44]: Text(89.18, 0.5, 'predicted class')
```

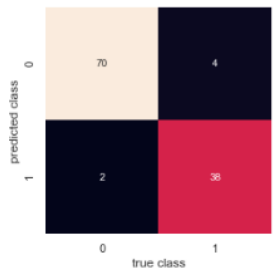


Figure 4.29:Confusion Matrix

5. Evaluation

5.1 Critical Analysis

With an accuracy score of about 94.7 percent, the Random Forest Classifier performed the best on the testing data. As a result, this would be an effective method for patients to detect cancer cells.

```
import sklearn.metrics as sm

In [67]:
# Get the accuracy of the test data
Y_prediction = rf_classifier.predict(X_test)
test_accuracy=sm.accuracy_score(Y_test, Y_prediction)
print('Testing Accuracy:',str(test_accuracy))

Testing Accuracy: 0.9473684210526315
```

Figure 5.1:Display the accuracy of the data set

figure 5.1 shows both the predictions data from the Random Forest Classifier and the actual values of the patient that shows that they diagnosed with the cancer or not.

```
In [71]:
# Compare the Predictions of Random Forest Classifier model and the actual classification of the patients
print("The Predictions of Random Forest Classifier model: \n\n", Y_prediction)
```

```
The Predictions of Random Forest Classifier model:

 [0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1
 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 1
 1 0 0]
```

```
In [72]:
print("\nThe actual classification of the patients: \n\n", Y_test)
```

```
The actual classification of the patients:

 28     0
163    0
123    1
361    1
549    0
      ..
414    0
515    1
186    1
3      0
261    0
Name: diagnosis, Length: 114, dtype: int64
```

Figure 5.2: Compare Random forest classifier prediction and the actual classification

5.2 k-Fold Cross Validation

To justify the optimal model for the predictions, we used k-fold cross validation. A well balance supervised learning model should have low variance and low bias values.

Variance = Measure of variability.
Variance = 0.01
Bias = Average prediction of the model - the correct value that is attempting to predict.

Bias = 0.947 – 0.953 = -0.006

```
In [73]:
# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator=rf_classifier, X=X_train, y=Y_train, cv=10)
print("Mean: ", accuracies.mean())
print("Standard Deviation: ", accuracies.std())
```

```
Mean:  0.953816425120773
Standard Deviation:  0.015495427946373158
```

Figure 5.3: Applying k-Fold Cross Validation

6. Conclusion

The random forest classification algorithm (RF) is used to lung cancer prediction. The accuracy score of the prediction is 94%. We use this classification model to lung cancer prediction in accurate rates. For the future prediction of lung cancers, the required fields and the required risk factors for the lung caners detection using the dataset.

7. References

[1] "Kaggle.com. [Online]," [Online]. Available: https://www.kaggle.com/datasets/prashanthpacchi/classification-datasetcancer-prediction.

[2] Borzenkova, "Types and Characteristics of Precipitation," *Hydrol. Cycle*, vol. II, 1999.

[3] A. E. Selase, D. Eunice, E. Agyimpomaa, D. D. Selasi, D. Melody, and N. Hakii, "Precipitation and Rainfall Types with Their Characteristic Features," *J. Nat. Sci. Res.*, vol. 5, no. 20, pp. 89–92, 2015.

[4] " Tutorialspoint.com. [Online]. Available:, "Classification Algorithms - Random Forest," [Online]. Available: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_%0Aclassification_algorithms_random_forest.htm.

8. Appendix

Video Demo link: https://drive.google.com/file/d/11jx1hjOjJ4K3OrIzyd0a-G50QIjvNJUE/view?usp=sharing

1. Report contribution

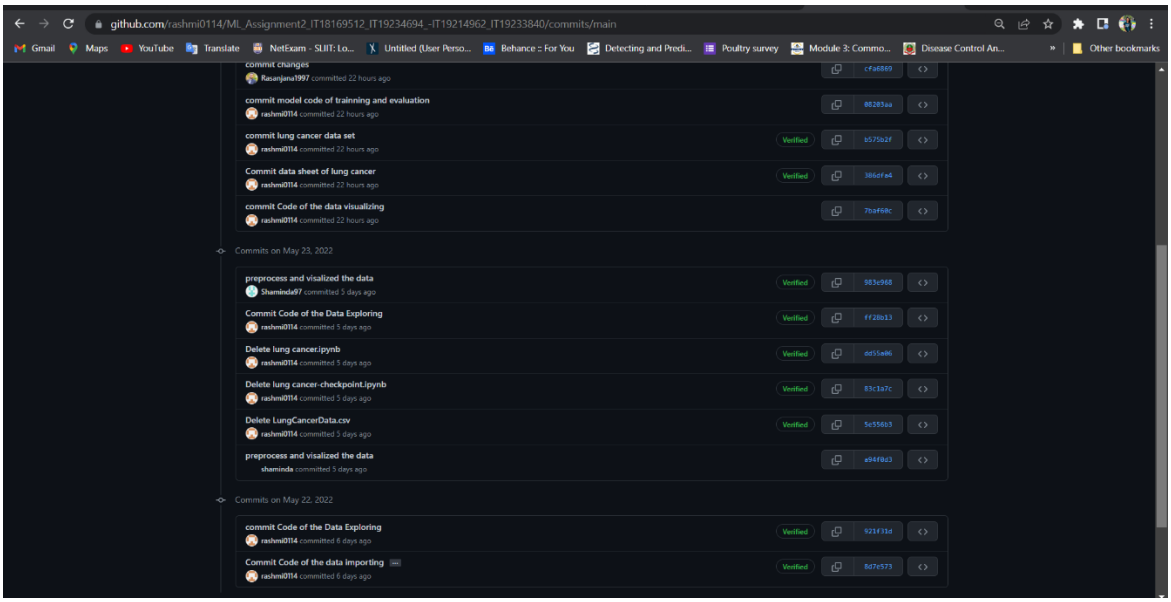
Workload	Contributor
Introduction	IT19214962
Dataset	IT19214962
Methodology	IT18160512
Implementation	IT19214962
Evaluation	IT19233840
Conclusion	IT19214962

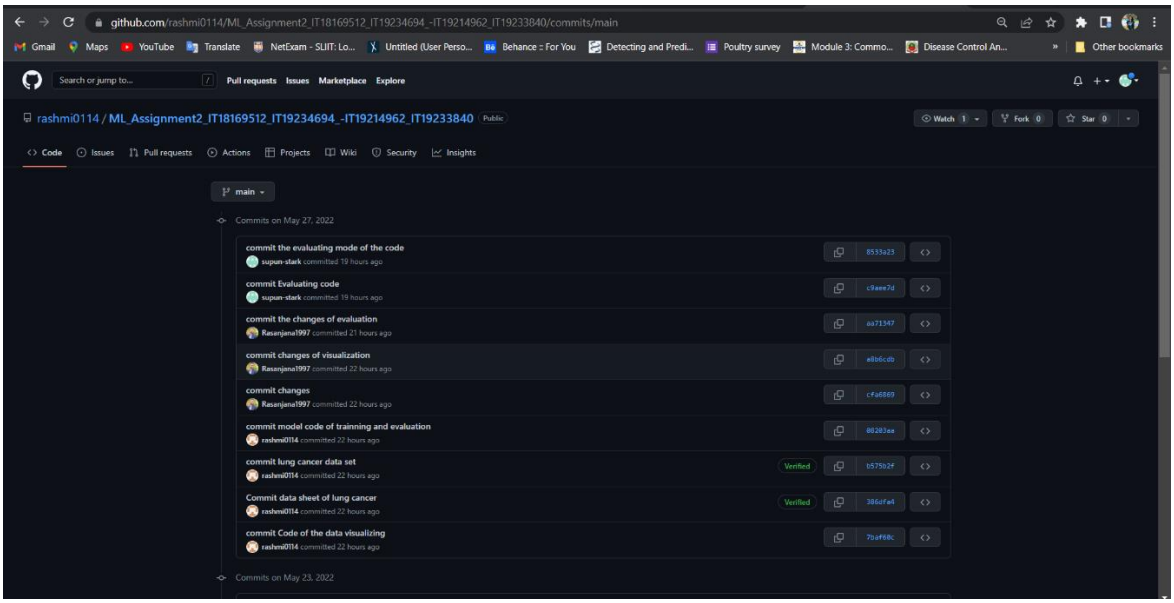
2. Code Contribution

Workload	Contributor
Importing Data	IT19234694
Exploring the data	IT19214962
Data Preprocessing	IT19233840
Visualizing the Data	IT19233840
Model Training	IT19214962
Evaluating the Mode	IT18160512

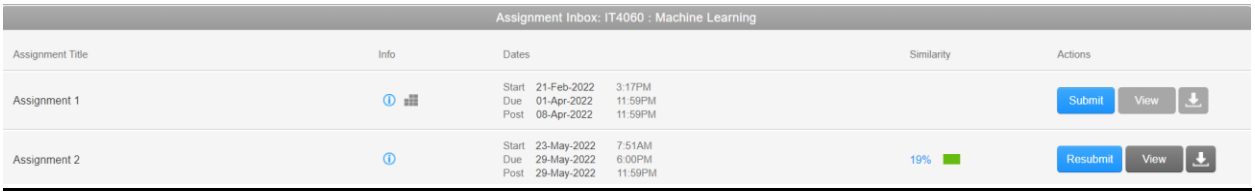
3. Individual Parts Contribution

Git commits





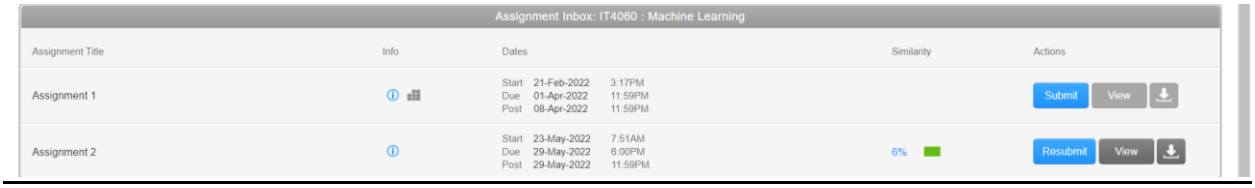
Similarity of the report – 19%



Assignment Inbox: IT4060 : Machine Learning

Assignment Title	Info	Dates	Similarity	Actions
Assignment 1		Start 21-Feb-2022 3:17PM Due 01-Apr-2022 11:59PM Post 08-Apr-2022 11:59PM		Submit View
Assignment 2		Start 23-May-2022 7:51AM Due 29-May-2022 6:00PM Post 29-May-2022 11:59PM	19%	Resubmit View

Similarity of the Code – 6%



Assignment Inbox: IT4060 : Machine Learning

Assignment Title	Info	Dates	Similarity	Actions
Assignment 1		Start 21-Feb-2022 3:17PM Due 01-Apr-2022 11:59PM Post 08-Apr-2022 11:59PM		Submit View
Assignment 2		Start 23-May-2022 7:51AM Due 29-May-2022 6:00PM Post 29-May-2022 11:59PM	6%	Resubmit View