Educate Elevate Enlightten

JSS Mahavidyapeetha

JSS Science And Technology University

(Established Under JSS Science and Technology University Act No. 43 of 2013)

(Formerly Known as SJCE)

JSS SCIENCE AND TECHNOLOGY UNIVERSITY MYSURU

CS745 Pattern Recognition

Event - IV

Topic: Indian Liver Patient Dataset (ILDP)

Submitted to:

Dr. Srinath S

Associate Professor

Department of Computer Science

JSS S&TU, Mysuru

Submitted by:

Sl. No	USN	NAME	Roll No
1	01JST18CS047	Kartik Nagaraj Nayak	17
2	01JST18CS166	Vidyasagar M S	51

CS-C SECTION, 7th Sem

1. Introduction

Pattern recognition is a data analysis method that uses machine learning algorithms to automatically recognize patterns and regularities in data. This data can be anything from text and images to sounds or other definable qualities. Pattern recognition systems can recognize familiar patterns quickly and accurately. They can also recognize and classify unfamiliar objects, recognize shapes and objects from different angles, and identify patterns and objects even if they're partially obscured. Classification is a process related to categorization, the process in which ideas and objects are recognized, differentiated and understood. Classification is the grouping of related facts into classes. Classification predictive modelling is the task of approximating a mapping function, f from input variables, X to discrete output variables, y. Classification belongs to the category of supervised learning where the targets are also provided with the input data. Two of the commonly used classifiers are the Naive Bayes and the k-Nearest Neighbours classifiers.

2. Aim

To understand and develop a simple Naive Bayes classifier and k-Nearest Neighbours classifier.

3. Objective

- Process Dataset.
- Develop a simple Naive Bayes Classifier.
- Develop a simple k-Nearest Neighbour Classifier.
- Analyse the models

4. Literature Survey

- Syed Hasan Adil, Mansoor Ebrahim, Kamran Raza, Syed Saad Azhar Ali, Manzoor Ahmed Hashmani " Liver Patient Classification using Logistic Regression" (2018)

  In this research paper, they have applied a machine learning approach to classify liver patients (i.e., Liver Patient or Not Liver Patient) using patient gender and laboratory medical test data. The labelled dataset was published on UCI machine learning repository as "Indian Liver Patient Records". The motivation behind this work is to apply a simple and less computational classification technique like Logistic Regression and compare its results with earlier results obtained on the same dataset by other researchers. The classification results of Logistic regression have proved its significance on this dataset by achieving better classification accuracy than NBC (Naïve Bayes Classifier), C4.5 (Decision Tree), SVM (Support Vector Machine), ANN (Artificial Neural Network), and KNN (K Nearest Neighbors) as presented in Ramana et al., research paper.

- Bendi Venkata Ramana, Prof. M.Surendra Prasad Babu "Liver Classification Using Modified Rotation Forest" International Journal of Engineering Research and Development ISSN: 2278-067X, Volume 1, Issue 6 (June 2012), PP.17-24

  Ensembling Classification techniques have been widely used in the medical field for accurate classification than an individual classifier. Modified Rotation Forest algorithm was proposed in this paper for accurate liver classification by analysing the combination of selected classification algorithm and feature selection technique. Selected classification algorithms were considered from each category of classification algorithms. The category of classification algorithms are Tree based, Statistical based, Neural Networks based, Rule based and Lazy learners. Modified Rotation Forest algorithm for UCI liver data set has multilayer perceptron classification algorithm and Random Subset feature selection technique and for INDIA liver data set has nearest neighbour with generalised distance function and correlation based feature selection technique.

5. Naive Bayes Classifier

It is one of the easiest classifiers to implement, it is mainly based on the bayes theorem given by:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

where A and B are events and P(B) ≠ 0.

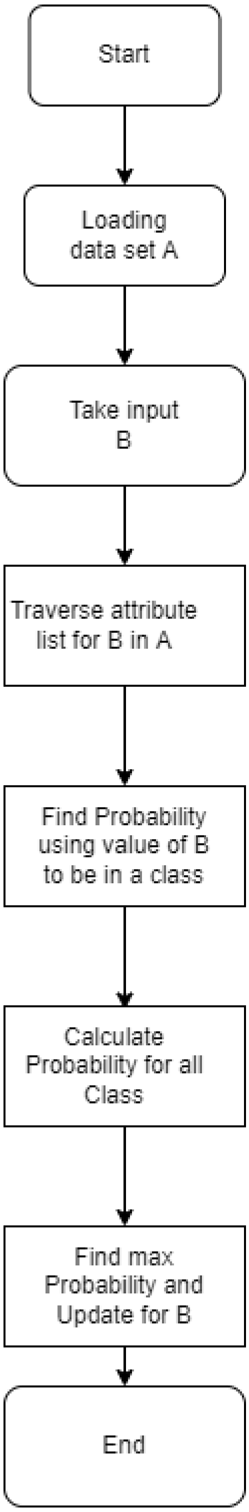
- Basically, we are trying to find the probability of event A, given that event B is true. Event B is also termed as evidence.
- P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

To apply the theorem for classification tasks, it can be rewritten using class and feature tags:

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

When there are multiple features we use X = (x1, x2, ... , xn).

A Naive Bayes classifier performs better compared to other classifiers when the assumption of independent features holds true. The classifier has the added advantage that it is simple and thus easy to implement, and that it requires a small amount of training data to estimate the test data which reduces the training time. However, the main limitation of the Naive Bayes classifier is the assumption of independent features, which is almost never the case in real life situations


Start
Loading
data set A
Take input
B
Traverse attribute
list for B in A
Find Probability
using value of B
to be in a class
Calculate
Probability for all
Class
Find max
Probability and
Update for B
End

.

6. k-Nearest Neighbours Classifier

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.
It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data. We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

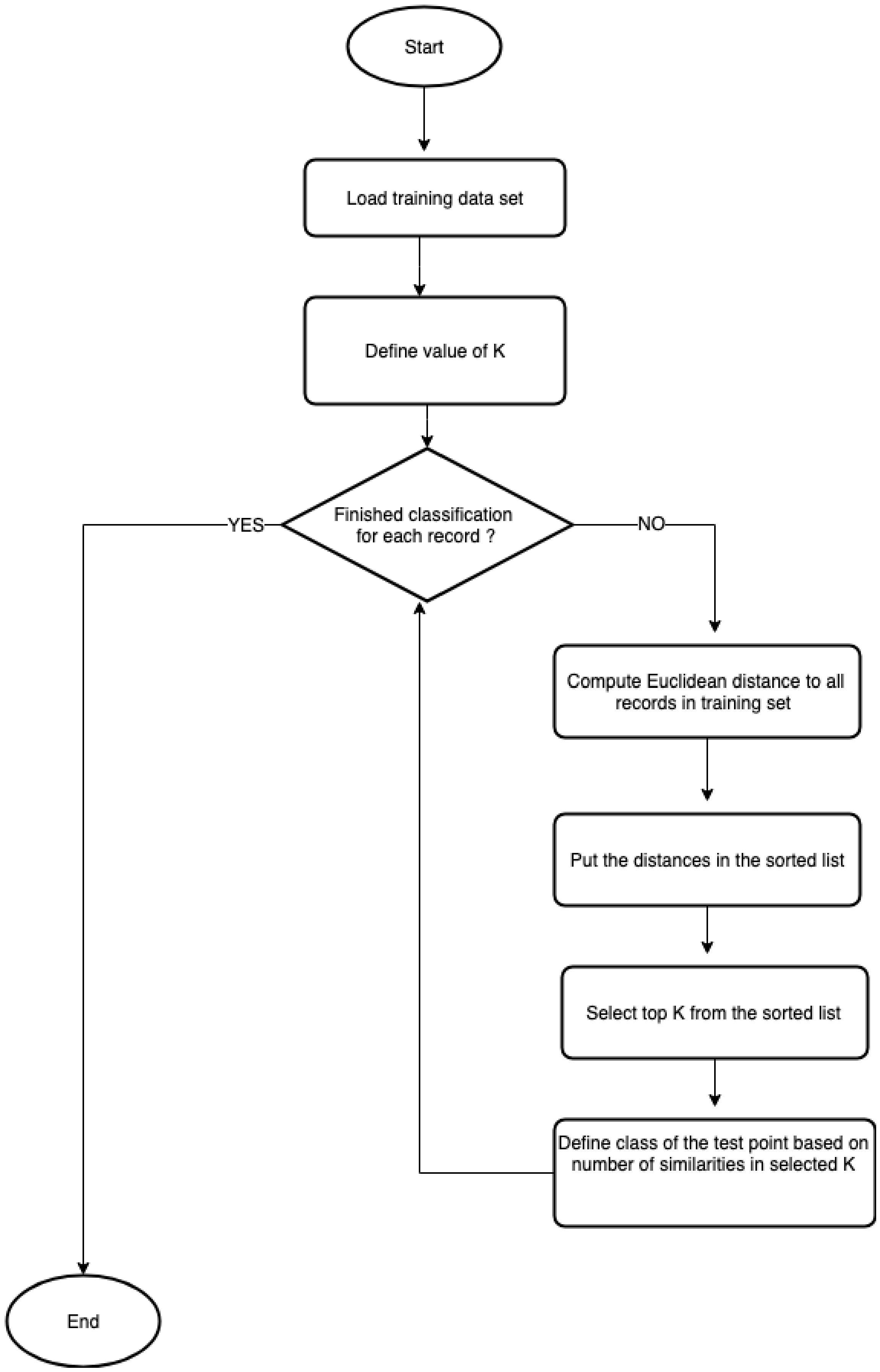
It is an easily implementable algorithm, it stores n-dimensional training data and when classification has to be performed it compares distance between the point that has to be classified and all the points in the dataset, the k nearest points are chosen and the majority class is assigned to the sample point.

Since the k-Nearest Neighbours classifier is a lazy learner, it does not need any training time at all. Since the classifier requires no training, new data can be easily added, which does not impact the accuracy of the classifier. Another advantage is that it is simple and easy to implement, which uses only two parameters, the value of k, and the distance function used.However, the k-Nearest Neighbours classifier suffers with large datasets, since it has to calculate the distance between the test sample and each point from the dataset, which requires a lot of time. The classifier also suffers when the dimensionality of the data is considerably high. The k-Nearest Neighbours classifier may also require feature scaling to provide fairly accurate results. The classifier is quite sensitive to noise in the dataset, and requires manually imputing missing values and removing outliers.

Algorithm

Let m be the number of training data samples. Let p be an unknown point.

1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).
   for i=0 to m:
   Calculate Euclidean distance d(arr[i], p).
1. Make the set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point.
2. Return the majority label among S.


Start
Load training data set
Define value of K
Finished classification
for each record ?
YES
NO
Compute Euclidean distance to all
records in training set
Put the distances in the sorted list
Select top K from the sorted list
Define class of the test point based on
number of similarities in selected K
End

7. Dataset

ILPD (Indian Liver Patient Dataset): This data set contains 416 liver patient records and 167 non liver patient records.The data set was collected from north east of Andhra Pradesh, India. Liver_disease is a class label used to divide into groups(liver patient or not). This data set contains 441 male patient records and 142 female patient records. Any patient whose age exceeded 89 is listed as being of age "90". It contains the following records

- **Age** of the patient
- **Gender** of the patient
- **Total Bilirubin**

  Bilirubin is an orange-yellow pigment that occurs normally when part of your red blood cells break down. A bilirubin test measures the amount of bilirubin in your blood. It’s used to help find the cause of health conditions like jaundice, anaemia, and liver disease.
- **Direct Bilirubin**

  Bilirubin attached by the liver to glucuronic acid, a glucose-derived acid, is called direct or conjugated, bilirubin. Bilirubin not attached to glucuronic acid is called indirect. All the bilirubin in your blood together is called total bilirubin.
- **Alkaline Phosphatase**

  Alkaline phosphatase (ALP) is an enzyme in a person’s blood that helps break down proteins. Using an ALP test, it is possible to measure how much of this enzyme is circulating in a person’s blood.
- **Alanine Aminotransferase**

  Alanine aminotransferase (ALT) is an enzyme found primarily in the liver and kidney. ALT is increased with liver damage and is used to screen for and/or monitor liver disease.
- **Aspartate Aminotransferase**

  AST (aspartate aminotransferase) is an enzyme that is found mostly in the liver, but also in muscles. When your liver is damaged, it releases AST into your bloodstream. An AST blood test measures the amount of AST in your blood. The test can help your health care provider diagnose liver damage or disease.
- **Total Proteins**

  Albumin and globulin are two types of protein in your body. The total protein test measures the total amount of albumin and globulin in your body.

- **Albumin**

  Albumin is a protein made by your liver. Albumin helps keep fluid in your bloodstream so it doesn't leak into other tissues. It also carries various substances throughout your body, including hormones, vitamins, and enzymes. Low albumin levels can indicate a problem with your liver or kidneys.

- **Albumin and Globulin Ratio**

  The Albumin to Globulin ratio (A:G) is the ratio of albumin present in serum in relation to the amount of globulin. The ratio can be interpreted only in light of the total protein concentration. Very generally speaking, the normal ratio in most species approximates 1:1.

- **Liver_Disease**

  This field is used to split the data into two sets (patient with liver disease, or no disease).

8. Implementation and Result

Indian Liver Patients

Import Required Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn import model_selection
from sklearn import metrics
from sklearn.model_selection import cross_val_score
%matplotlib inline
```

Load Dataset

In [2]:

```
df = pd.read_csv('indian_liver_patient.csv')
df.head()
```

Out[2]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alanine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Liver_disease
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

Pre Processing

In [3]:
```
df.describe()
```

Out[3]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alanine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Liver_disease
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	579.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.947064	1.286449
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.319592	0.452490
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	1.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	1.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	1.000000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	1.100000	2.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	2.000000

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         583 non-null    int64
 1   Gender                      583 non-null    object
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64
 5   Alanine_Aminotransferase    583 non-null    int64
 6   Aspartate_Aminotransferase  583 non-null    int64
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  579 non-null    float64
 10  Liver_disease               583 non-null    int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Age                           0
Gender                        0
Total_Bilirubin               0
Direct_Bilirubin              0
Alkaline_Phosphotase          0
Alanine_Aminotransferase      0
Aspartate_Aminotransferase    0
Total_Protiens                0
Albumin                       0
Albumin_and_Globulin_Ratio    4
Liver_disease                 0
dtype: int64
```

In [6]:
```
df = df.dropna()
# Changing the values in "Liver_Disease" column
df['Liver_disease'] = df['Liver_disease'] - 1
# Converting Gender column into categorical data
LabelEncoder = LabelEncoder()
df['Is_male'] = LabelEncoder.fit_transform(df['Gender'])
df = df.drop(columns='Gender')
```

In [7]:
```
df.head()
```

Out[7]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alanine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Liver_disease	Is_male
0	65	0.7	0.1	187	16	18	6.8	3.3	0.90	0	0
1	62	10.9	5.5	699	64	100	7.5	3.2	0.74	0	1
2	62	7.3	4.1	490	60	68	7.0	3.3	0.89	0	1
3	58	1.0	0.4	182	14	20	6.8	3.4	1.00	0	1
4	72	3.9	2.0	195	27	59	7.3	2.4	0.40	0	1

In [8]:
```
X = df[['Age', 'Total_Bilirubin',
        'Direct_Bilirubin',
        'Alkaline_Phosphotase',
        'Alanine_Aminotransferase', 'Aspartate_Aminotransferase',
       'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio', 'Is_male']]
y = df['Liver_disease']
```

In [9]:
```
print ('Total Unhealthy Livers :  {} and its percentage is {} %'.format(df.Liver_disease.value_counts()[0], round(df.Liver_disease.value_counts()[0]/df.Liver_disease.value_counts().sum()*100,2)))
print ('Total Healthy Livers :  {} and its percentage is {} %'.format(df.Liver_disease.value_counts()[1], round(df.Liver_disease.value_counts()[1]/df.Liver_disease.value_counts().sum()*100,2)) )
```

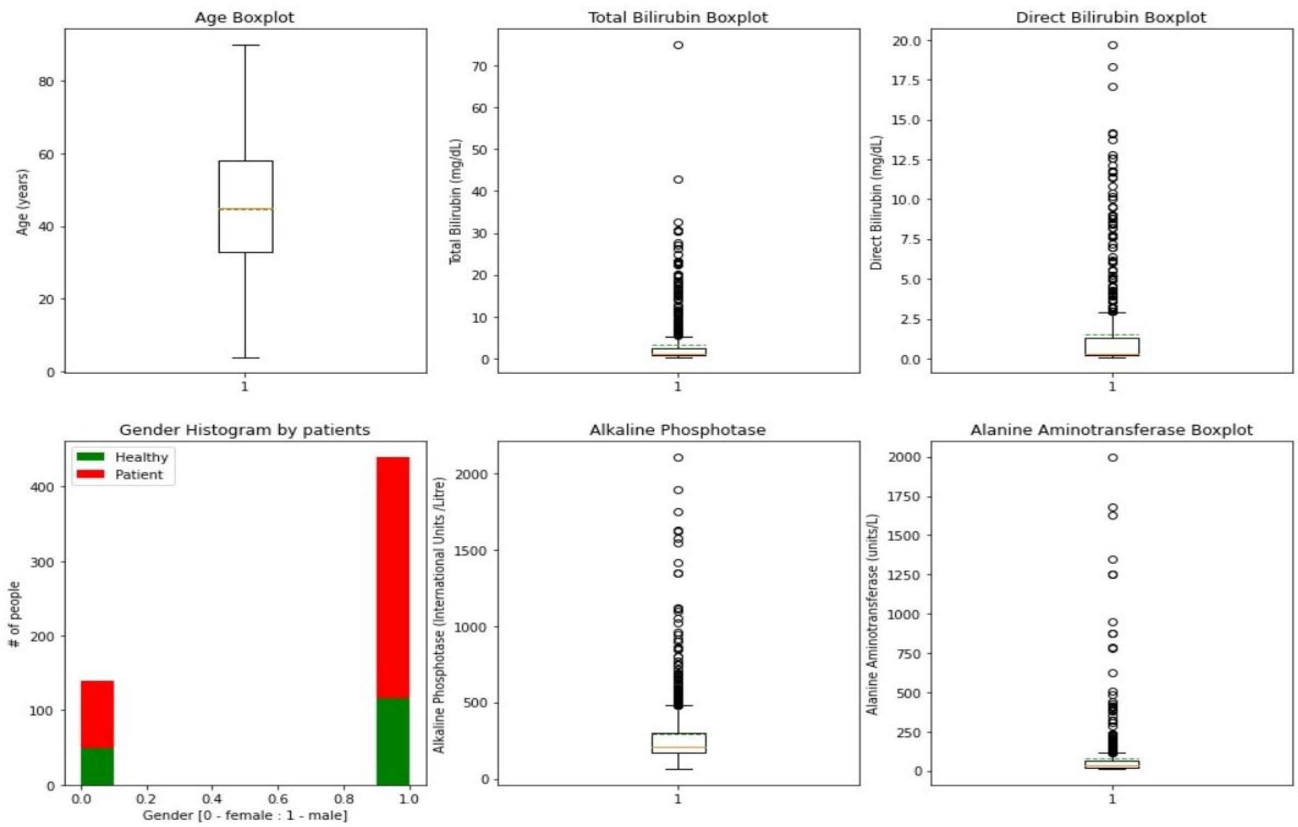
```
Total Unhealthy Livers :  414 and its percentage is 71.5 %
Total Healthy Livers :  165 and its percentage is 28.5 %
```

In [10]:
```
df.skew(axis = 0, skipna = True)
```

Out[10]:
```
Age                           -0.033591
Total_Bilirubin                4.890768
Direct_Bilirubin               3.199163
Alkaline_Phosphotase           3.753502
Alanine_Aminotransferase       6.527575
Aspartate_Aminotransferase    10.512251
Total_Protiens                -0.292433
Albumin                       -0.048516
Albumin_and_Globulin_Ratio     0.992299
Liver_disease                  0.955179
Is_male                       -1.209212
dtype: float64
```

In [11]:
```python
# Plotting the box plots
plt.figure(figsize=[16,12])

plt.subplot(231)
plt.boxplot(x = X['Age'], showmeans = True, meanline = True)
plt.title('Age Boxplot')
plt.ylabel('Age (years)')

plt.subplot(232)
plt.boxplot(X['Total_Bilirubin'], showmeans = True, meanline = True)
plt.title('Total Bilirubin Boxplot')
plt.ylabel('Total Bilirubin (mg/dL)')

plt.subplot(233)
plt.boxplot(X['Direct_Bilirubin'], showmeans = True, meanline = True)
plt.title('Direct Bilirubin Boxplot')
plt.ylabel('Direct Bilirubin (mg/dL)')

plt.subplot(234)
plt.hist(x = [X[y==1]['Is_male'], X[y ==0]['Is_male']],
         stacked=True, color = ['g','r'],label = ['Healthy','Patient'])
plt.title('Gender Histogram by patients')
plt.xlabel('Gender [0 - female : 1 - male]')
plt.ylabel('# of people')
plt.legend()

plt.subplot(235)
plt.boxplot(x = X['Alkaline_Phosphotase'], showmeans = True, meanline = True)
plt.title('Alkaline Phosphotase')
plt.ylabel('Alkaline Phosphotase (International Units /Litre)')

plt.subplot(236)
plt.boxplot(X['Alanine_Aminotransferase'], showmeans = True, meanline = True)
plt.title('Alanine Aminotransferase Boxplot')
plt.ylabel('Alanine Aminotransferase (units/L)')
```

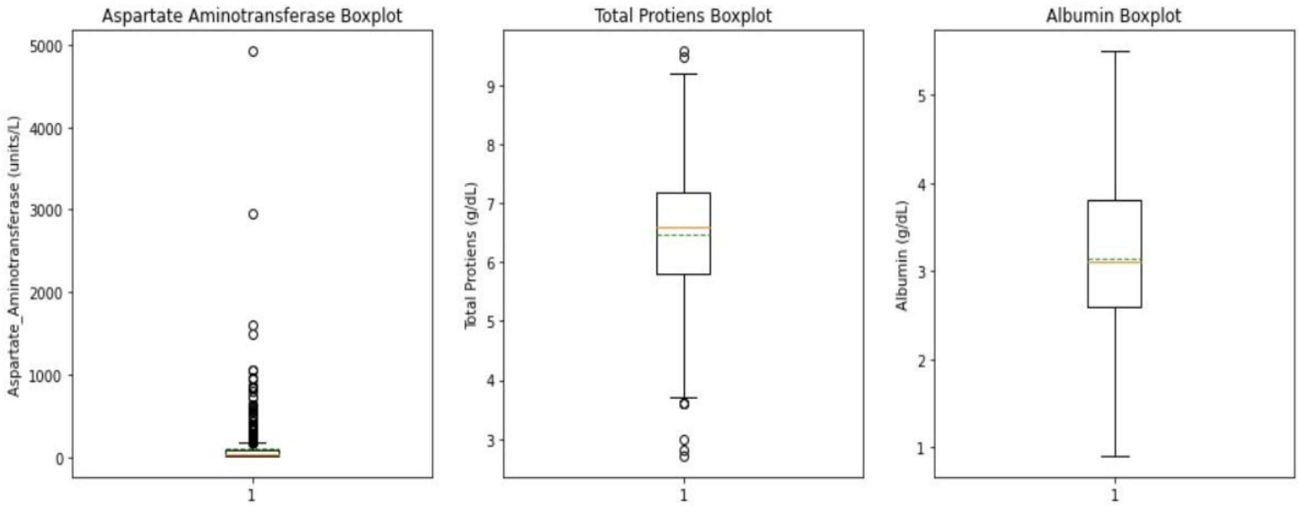
Out[11]: Text(0, 0.5, 'Alanine Aminotransferase (units/L)')



In [12]:

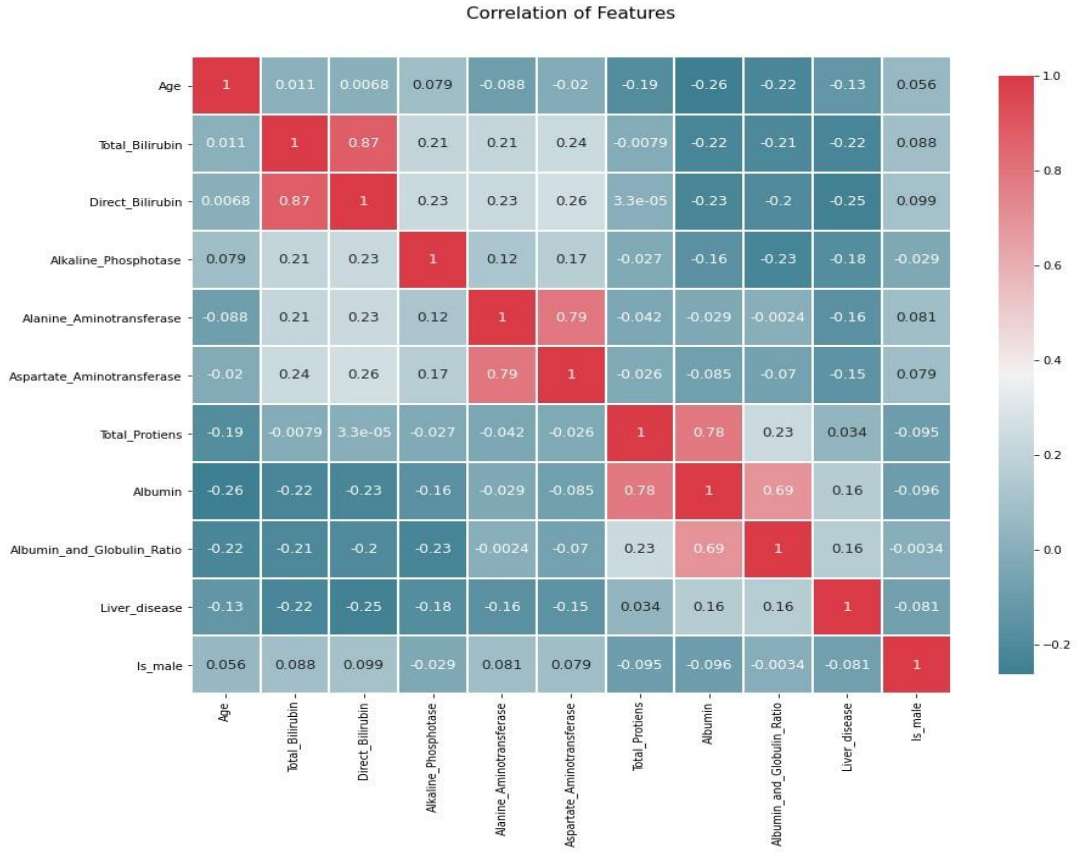
```
plt.figure(figsize=[16,12])
plt.subplot(231)
plt.boxplot(X['Aspartate_Aminotransferase'], showmeans = True, meanline = True)
plt.title('Aspartate Aminotransferase Boxplot')
plt.ylabel('Aspartate_Aminotransferase (units/L)')


plt.subplot(232)
plt.boxplot(X['Total_Protiens'], showmeans = True, meanline = True)
plt.title('Total Protiens Boxplot')
plt.ylabel('Total Protiens (g/dL)')

plt.subplot(233)
plt.boxplot(X['Albumin'], showmeans = True, meanline = True)
plt.title('Albumin Boxplot')
plt.ylabel('Albumin (g/dL)')
```

Out[12]: Text(0, 0.5, 'Albumin (g/dL)')



In [13]:

```
def correlation_heatmap(df):
    _ , ax = plt.subplots(figsize =(14, 12))
    colormap = sns.diverging_palette(220, 10, as_cmap = True)

    _ = sns.heatmap(
        df.corr(),
        cmap = colormap,
        square=True,
        cbar_kws={'shrink':.9 },
        ax=ax,
        annot=True,
        linewidths=0.1,vmax=1.0, linecolor='white',
        annot_kws={'fontsize':12 }
    )

    plt.title('Correlation of Features', y=1.05, size=15)

correlation_heatmap(df)
```




```
In [14]:
# normalise the data
from sklearn import preprocessing
X_scaler = preprocessing.normalize(X)
```

```
In [15]:
# Splitting the data
X_train, X_test, y_train, y_test = model_selection.train_test_split(X_scaler, y, random_state = 17,test_size=0.2)

print("Train Shape: {}".format(X_train.shape))
print("Test Shape: {}".format(X_test.shape))
```

```
Train Shape: (463, 10)
Test Shape: (116, 10)
```

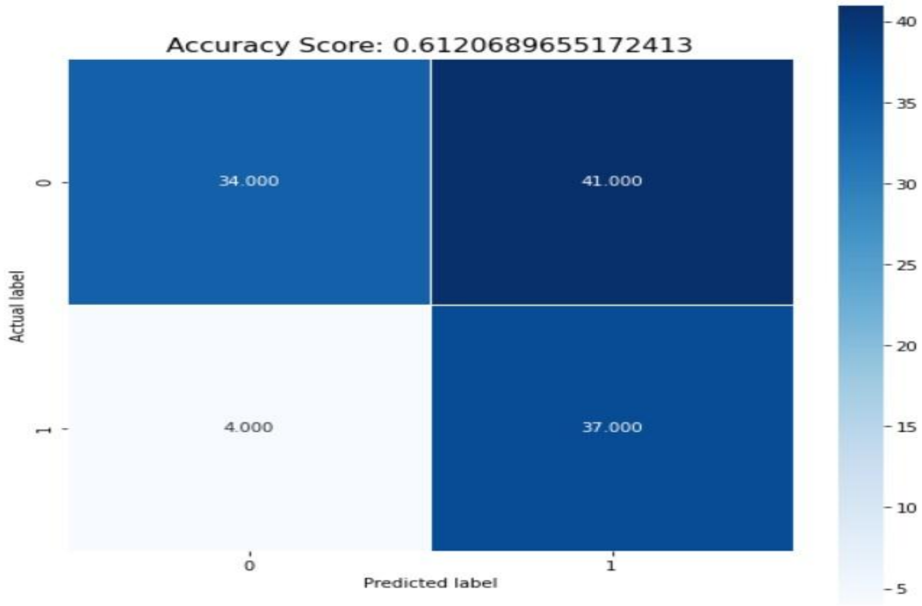
Model

Naive Bayes Model

```
In [16]:
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,y_train)
y_pred_nb = nb.predict(X_test)
```

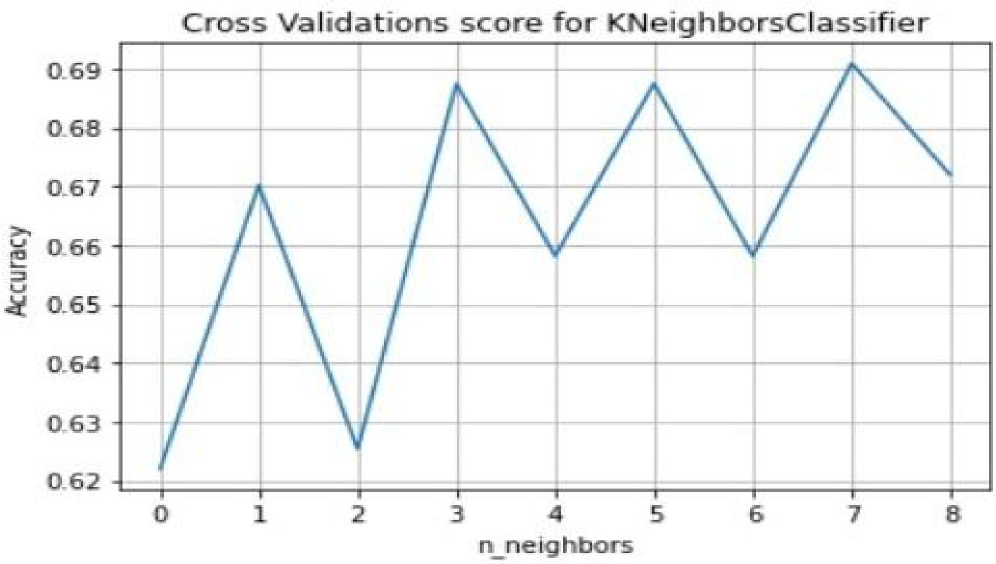
```
In [17]:
score = nb.score(X_test, y_test)
cm = metrics.confusion_matrix(y_test, y_pred_nb)
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15)
```

Out[17]: Text(0.5, 1.0, 'Accuracy Score: 0.6120689655172413')



K-Nearest Neighbour Classifier

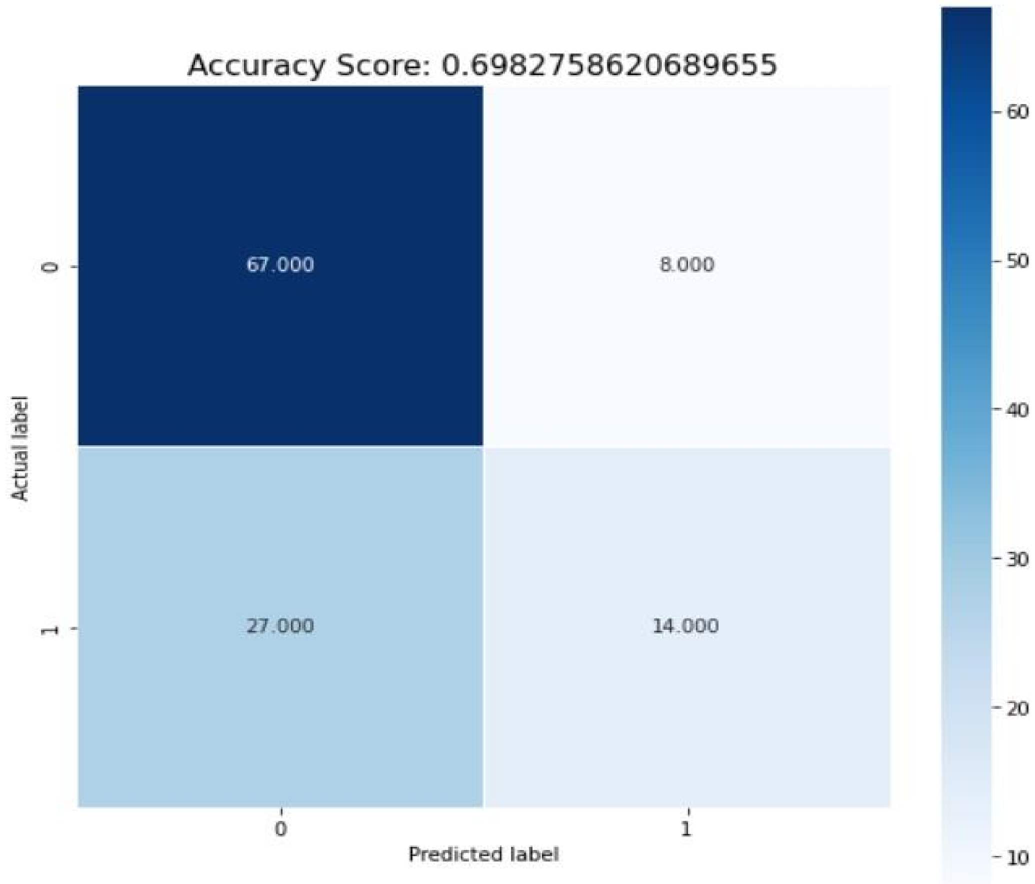
In [18]:
```
# KNN Model
from sklearn.neighbors import KNeighborsClassifier
hist = []
for i in range(1,10):
    clf = KNeighborsClassifier(n_neighbors=i)
    cross_val = cross_val_score(clf, X_scaler, y, cv=5)
    hist.append(np.mean(cross_val))
plt.plot(hist)
plt.title('Cross Validations score for KNeighborsClassifier')
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.grid()
plt.show()
```



In [19]:
```
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train,y_train)
y_pred_knn = knn.predict(X_test)
```

In [20]:
```
score = knn.score(X_test, y_test)
cm = metrics.confusion_matrix(y_test, y_pred_knn)
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15)
```

Out[20]:
```
Text(0.5, 1.0, 'Accuracy Score: 0.6982758620689655')
```



In [21]:

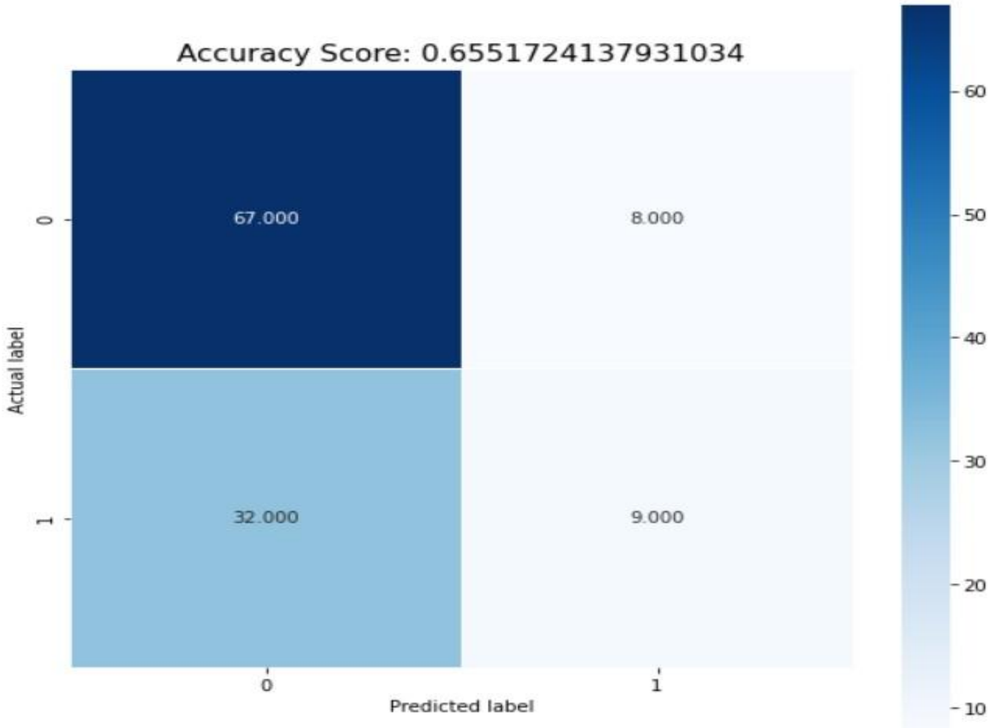
```
knn2 = KNeighborsClassifier(n_neighbors = 7)
knn2.fit(X_train,y_train)
y_pred_knn = knn2.predict(X_test)
```

In [22]:

```
score = knn2.score(X_test, y_test)
cm = metrics.confusion_matrix(y_test, y_pred_knn)
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15)
```

Out[22]:

```
Text(0.5, 1.0, 'Accuracy Score: 0.6551724137931034')
```



9. Conclusion

Out of all the machine learning algorithms KNN algorithm has easily been the simplest to pick up. Despite its simplicity, it has proven to be incredibly effective at certain tasks. From the above implementation we can conclude that KNN algorithm with k as 5 best classifies the data into Liver disease or not. We can further compare this algorithm with other machine learning classifiers like decision tree, adaboost classifier etc.

10. Reference

[1] https://data.world/uci/ilpd-indian-liver-patient-dataset

[2] https://www.geeksforgeeks.org/naive-bayes-classifiers/

[3] https://www.geeksforgeeks.org/k-nearest-neighbours/

[4] https://ieeexplore.ieee.org/document/8510581

[5] http://www.ijerd.com/paper/vol1-issue6/C0161724.pdf