In [1]:

```
from pandas import Series,DataFrame

# import libs
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# machine learning
import sklearn
from sklearn import preprocessing
from scipy.stats import pearsonr

# machine learning  - supervised
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import accuracy_score

# machine learning  - unsupervised
from sklearn import decomposition
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
# visualization and plotting
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
%matplotlib qt
import warnings
warnings.filterwarnings('ignore')
```

In [5]:

```
df=pd.read_csv("indian_liver_patient.csv")
df.head()
df.describe()
df.describe(include=['O'])
df.info()
df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
Age                           583 non-null int64
Gender                        583 non-null object
Total_Bilirubin               583 non-null float64
Direct_Bilirubin              583 non-null float64
Alkaline_Phosphotase          583 non-null int64
Alamine_Aminotransferase      583 non-null int64
Aspartate_Aminotransferase    583 non-null int64
Total_Protiens                583 non-null float64
Albumin                       583 non-null float64
Albumin_and_Globulin_Ratio    579 non-null float64
Dataset                       583 non-null int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

Out[5]:

```
(583, 11)
```

In [6]:

```
df['Dataset'].value_counts()
```

Out[6]:

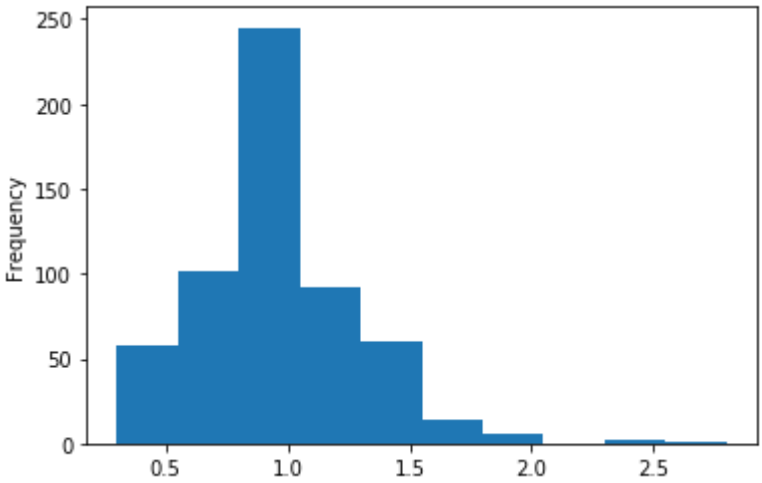
```
1    416
2    167
Name: Dataset, dtype: int64
```

In [7]:

```
df[df["Albumin_and_Globulin_Ratio"].isnull()]
df["Albumin_and_Globulin_Ratio"].plot(kind='hist')
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19c9f3c57b8>
```



In [9]:

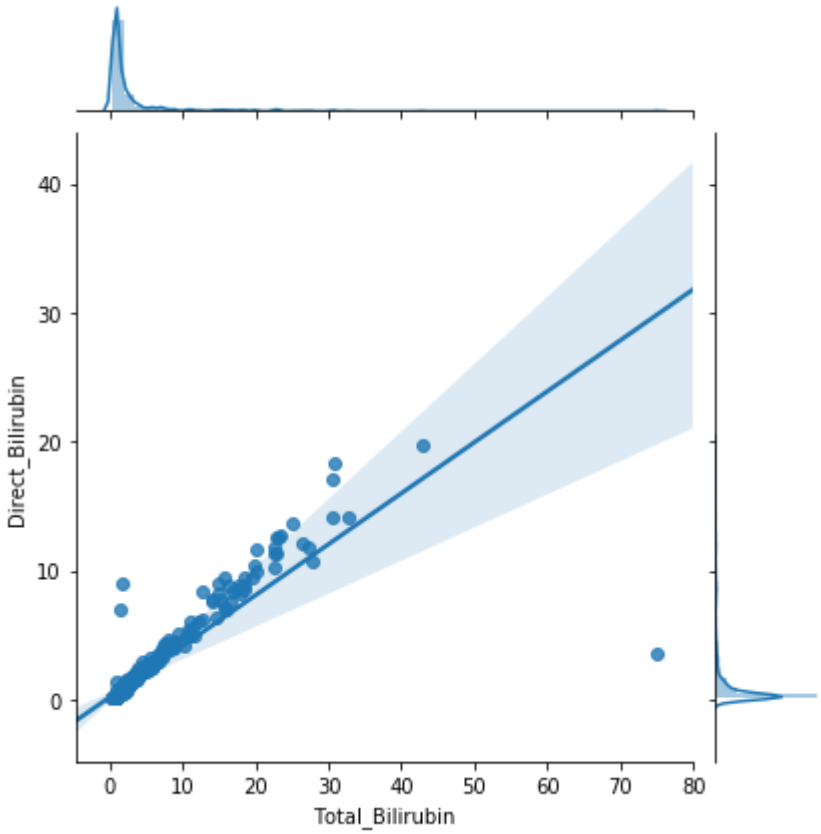
```
df.Albumin_and_Globulin_Ratio.fillna(df['Albumin_and_Globulin_Ratio'].median(), inplace=Tru
```

In [10]:

```
# encode gender

le = preprocessing.LabelEncoder()
le.fit(df.Gender.unique())
df['Gender_Encoded'] = le.transform(df.Gender)
df.drop(['Gender'], axis=1, inplace=True)
```

In [11]:

```
# correlation plots
g = sns.jointplot("Total_Bilirubin", "Direct_Bilirubin", data=df, kind="reg")
```



In [12]:

```
# calculate correlation coefficients for two variables
print(pearsonr(df['Total_Bilirubin'], df['Direct_Bilirubin']))
```

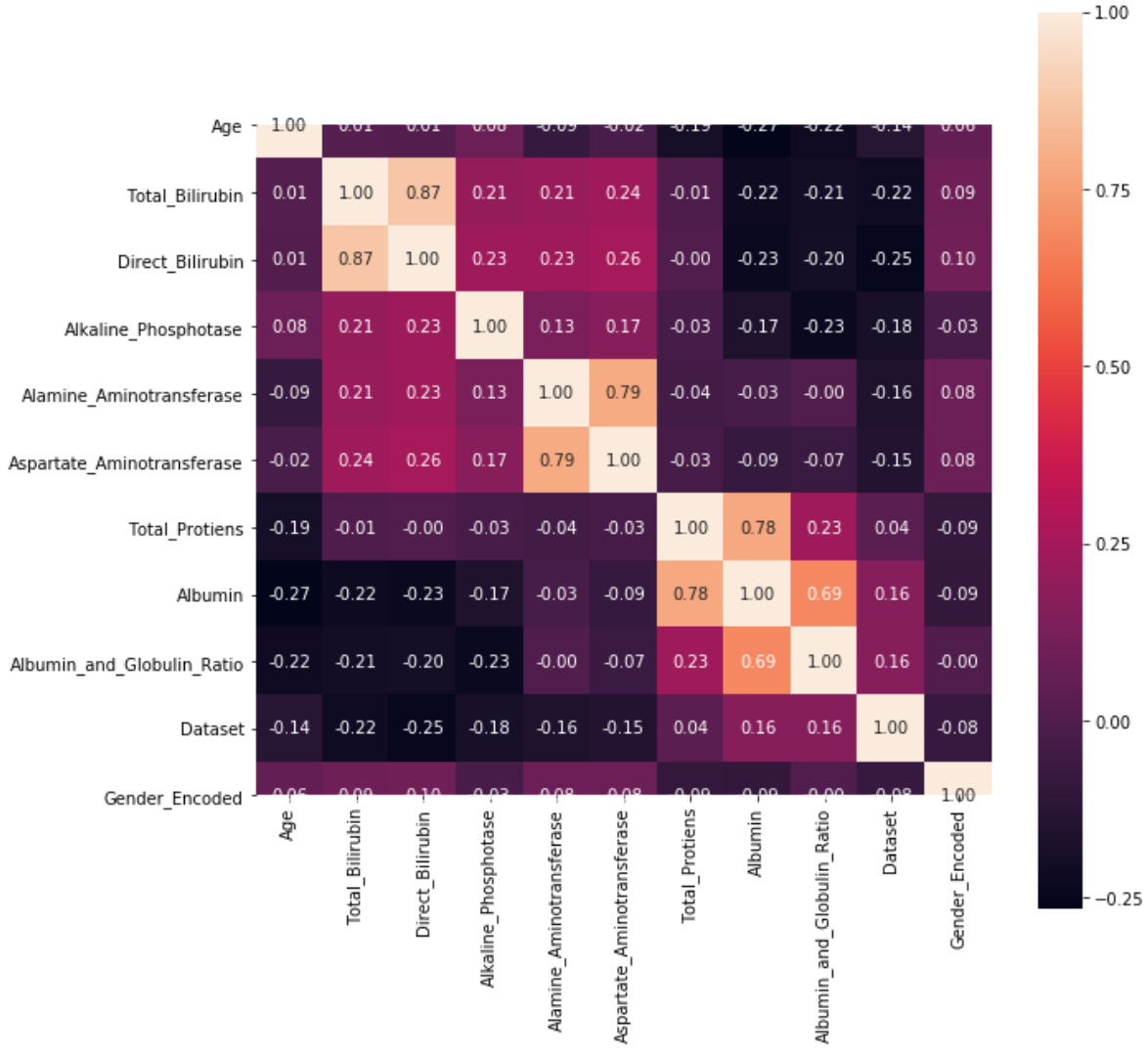
```
(0.8746179301164123, 7.873553178042517e-185)
```

In [13]:

```
# calculate correlation coefficients for all dataset
correlations = df.corr()
```

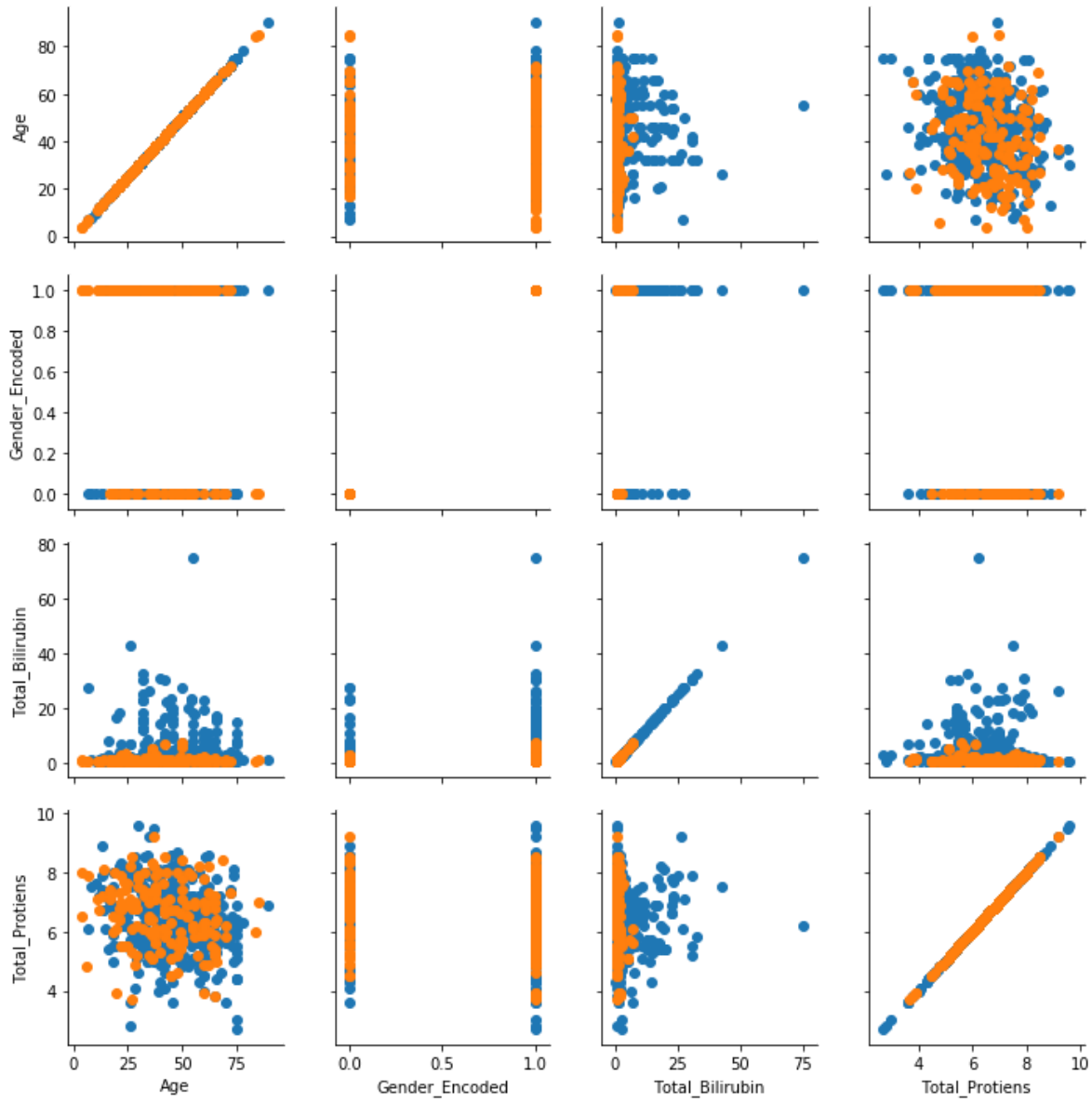
In [14]:

```
# and visualize
plt.figure(figsize=(10, 10))
g = sns.heatmap(correlations, cbar = True, square = True, annot=True, fmt= '.2f', annot_kws
```



In [16]:

```
# based on correlation, you can exclude some highly correlated features


# pair grid allows to visualize multi-dimensional datasets
g = sns.PairGrid(df, hue="Dataset", vars=['Age','Gender_Encoded','Total_Bilirubin','Total_P
g.map(plt.scatter)
plt.show()
```



In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(label='count',x='Dataset',data=df)
sns.catplot(data=df,y='Age',x='Gender_Encoded',hue='Dataset',jitter=0.4)
sns.jointplot("Total_Bilirubin", "Direct_Bilirubin", data=df, kind="reg")
sns.jointplot("Aspartate_Aminotransferase", "Alamine_Aminotransferase", data=df, kind="reg"
sns.jointplot("Total_Protiens", "Albumin", data=df, kind="reg")
df.corr()
```

In [18]:

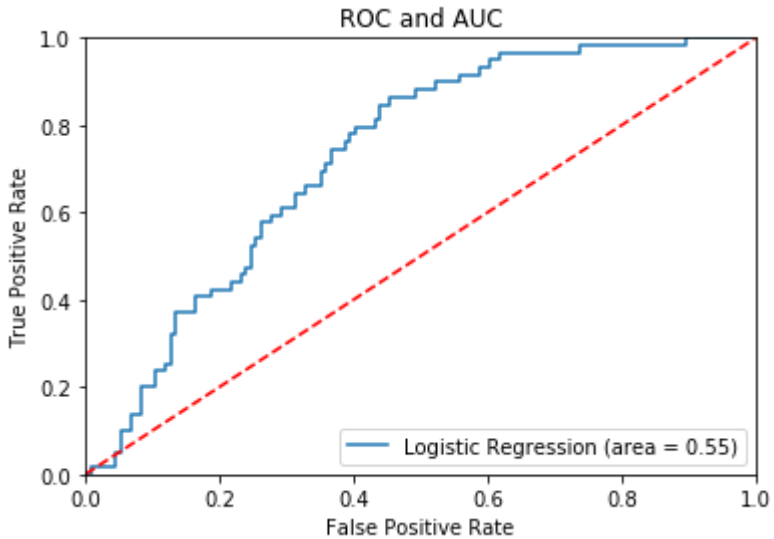
```
#Results of Analysis:
#Age and Gender affect the occurence of disease.
#Some features are directly correlated like Total_Bilirubin and Direct_Bilirubin, Aspartate
#Male has more the no of liver disease than female.
#Since gender is categorical we need to convert it to numeric data.
df['Dataset']=df['Dataset'].map({1:0,2:1}).astype(int)
```

In [19]:

```
X = df.drop(['Gender_Encoded','Dataset','Direct_Bilirubin','Aspartate_Aminotransferase'], a
y = df['Dataset']
xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=0.33, random_state=101)
```

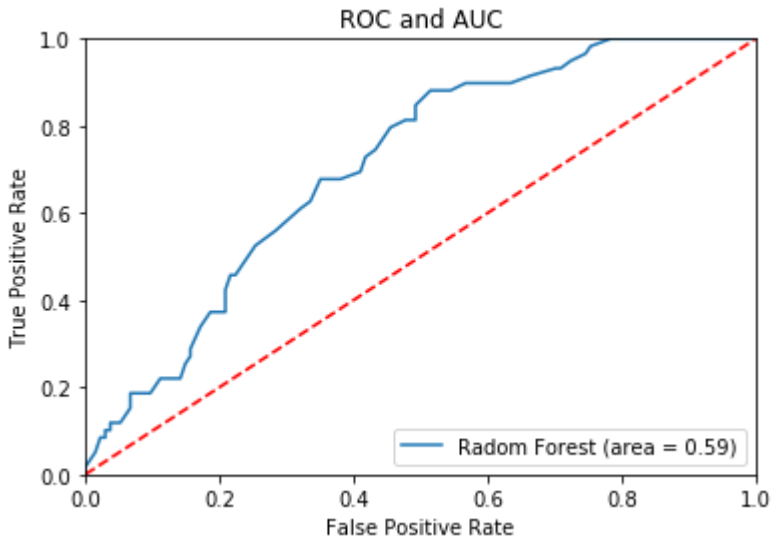
In [20]:

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(xTrain,yTrain)
#Finding the performance of model

lg_pred = logmodel.predict(xTest)
#confusion matrix
from sklearn.metrics import confusion_matrix

confusion_matrix(yTest,lg_pred)
cmlog= confusion_matrix(yTest, lg_pred)
cmlog[1,1]
cmlog.sum()
acculog= (cmlog[0,0]+cmlog[1,1])/cmlog.sum()*100
round(acculog)
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

lg_roc_auc = roc_auc_score(yTest,logmodel.predict(xTest))
round(lg_roc_auc*100)

#Probability
Lgprob = logmodel.predict_proba(xTest)

lgfpr,lgtpr,thresholds = roc_curve(yTest,logmodel.predict_proba(xTest)[:,1])

#Plot ROC and AUC

plt.figure()
plt.plot(lgfpr,lgtpr,label='Logistic Regression (area = %0.2f)' %lg_roc_auc)
plt.plot([0,1],[0,1],'r--') #r means color red and -- means style of line
plt.xlim([0.0,1.0]) #limit of x axis
plt.ylim([0.0,1.0]) #limit of y axis
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC and AUC')
plt.legend(loc="lower right") #location of Legend by default is left top
plt.show()
```



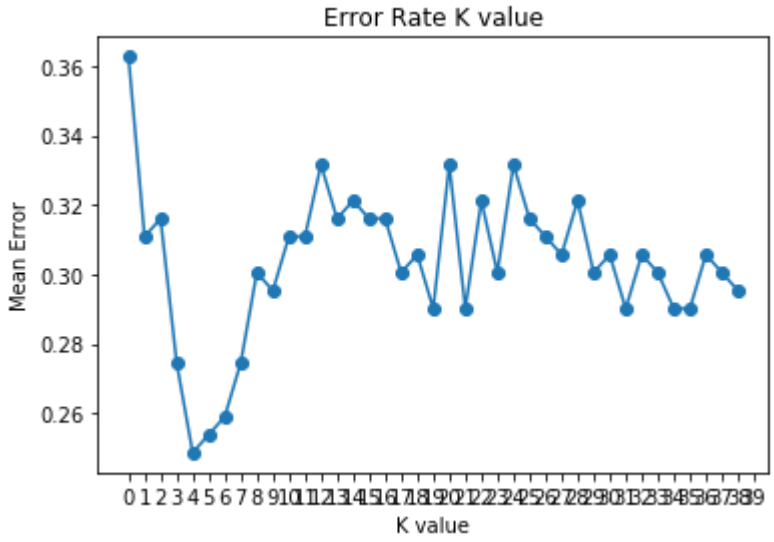
In [21]:

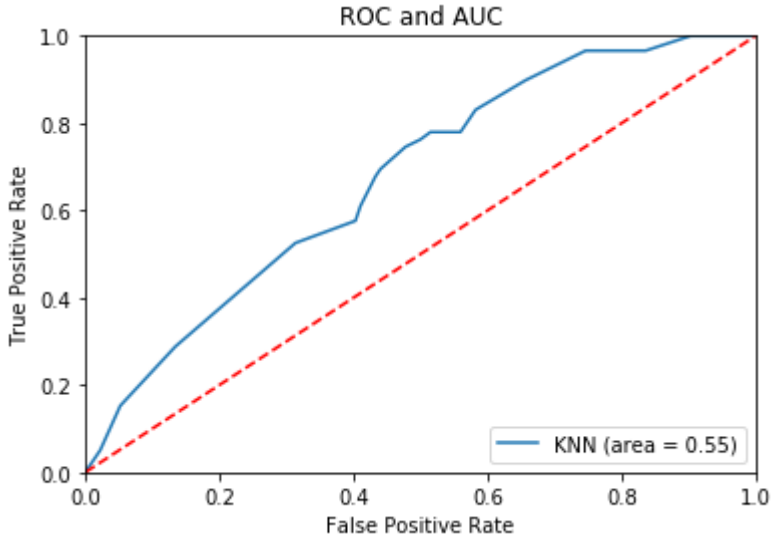
```
#Random forest
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier(n_estimators= 100, oob_score = True, random_state=12

random_forest.fit(xTrain,yTrain)
Y_prediction = random_forest.predict(xTest)
random_forest.score(xTrain,yTrain)
#Random Forest confusion matrix
from sklearn.metrics import confusion_matrix

confusion_matrix(yTest, Y_prediction)

rf_roc_auc = roc_auc_score(yTest,random_forest.predict(xTest))
round(rf_roc_auc*100)

#Random forest Probability
rfprob = random_forest.predict_proba(xTest)

rffpr,rftpr,thresholds = roc_curve(yTest,random_forest.predict_proba(xTest)[:,1])
plt.figure()
plt.plot(rffpr,rftpr,label='Radom Forest (area = %0.2f)' %rf_roc_auc)
plt.plot([0,1],[0,1],'r--') #r means color red and -- means style of line
plt.xlim([0.0,1.0]) #limit of x axis
plt.ylim([0.0,1.0]) #limit of y axis
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC and AUC')
plt.legend(loc="lower right") #location of Legend by default is left top
plt.show()
```



In [24]:

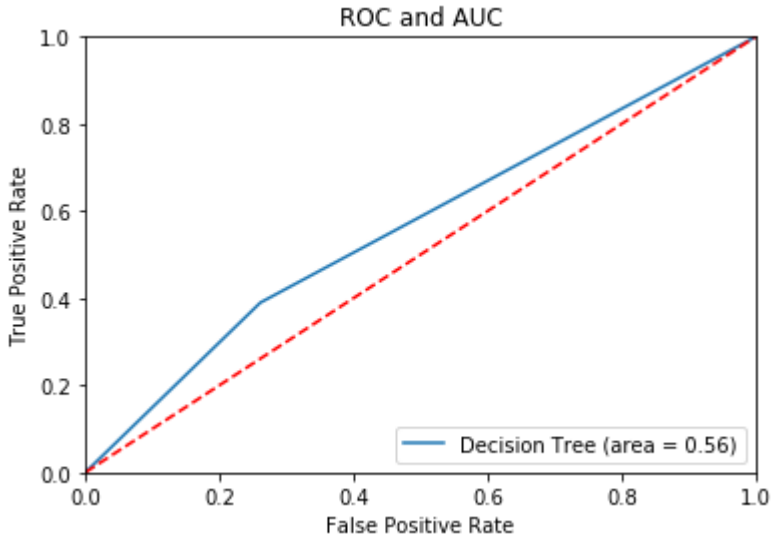
```
'''KNN'''

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=39)
knn.fit(xTrain,yTrain)

Y_pred_knn = knn.predict(xTest)
confusion_matrix(yTest,Y_pred_knn)
#tuning KNN
error = []
#calculating error for k values between 1 and 40
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(xTrain,yTrain)
    pred_i = knn.predict(xTest)
    error.append(np.mean(pred_i != yTest))

plt.figure()
plt.plot(error,marker='o')
plt.xticks(np.arange(0,40,step=1))
plt.title('Error Rate K value')
plt.xlabel('K value')
plt.ylabel('Mean Error')
'''n_neighbors = 22 because it is the lowest error
or 39 because it is giving the highest AUC'''

#Plot ROC and AUC
knn_roc_auc = roc_auc_score(yTest,knn.predict(xTest))
round(knn_roc_auc*100)
knnprob = knn.predict_proba(xTest)
knnfpr,knntpr,thresholds = roc_curve(yTest,knn.predict_proba(xTest)[:,1])

plt.figure()
plt.plot(knnfpr,knntpr,label='KNN (area = %0.2f)' %knn_roc_auc)
plt.plot([0,1],[0,1],'r--') #r means color red and -- means style of line
plt.xlim([0.0,1.0]) #limit of x axis
plt.ylim([0.0,1.0]) #limit of y axis
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC and AUC')
plt.legend(loc="lower right") #location of Legend by default is left top
plt.show()
```


Error Rate K value
Mean Error
K value
0.36
0.34
0.32
0.30
0.28
0.26


ROC and AUC
True Positive Rate
False Positive Rate
KNN (area = 0.55)
0.0
0.2
0.4
0.6
0.8
1.0

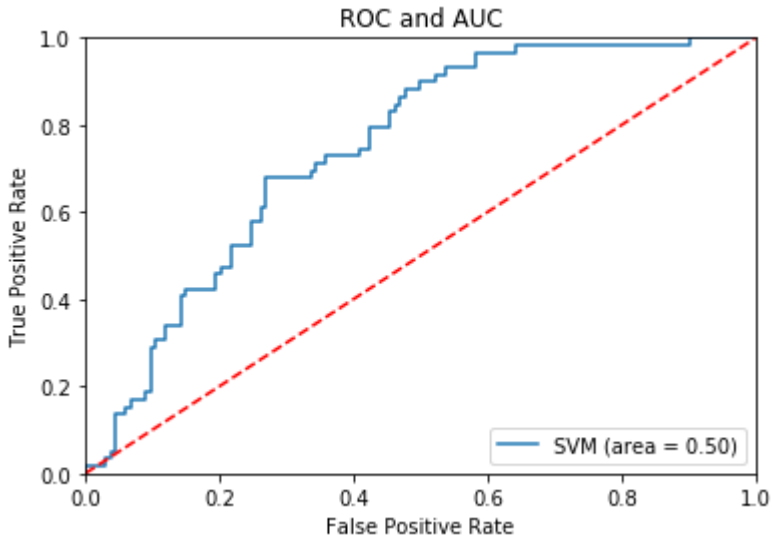
In [25]:

```
#decision tree
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(xTrain,yTrain)
Y_Pred_DT = decision_tree.predict(xTest)
confusion_matrix(yTest, Y_Pred_DT)


dtrob = decision_tree.predict_proba(xTest)
dt_roc_auc = roc_auc_score(yTest,decision_tree.predict(xTest))
dtfpr,dttpr,thresholds = roc_curve(yTest,decision_tree.predict_proba(xTest)[:,1])
plt.figure()
plt.plot(dtfpr,dttpr,label='Decision Tree (area = %0.2f)' %dt_roc_auc)
plt.plot([0,1],[0,1],'r--') #r means color red and -- means style of line
plt.xlim([0.0,1.0]) #limit of x axis
plt.ylim([0.0,1.0]) #limit of y axis
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC and AUC')
plt.legend(loc="lower right") #location of Legend by default is left top
plt.show()
```



In [26]:

```
#support Vector Machines (SVM) ensemble model faster

from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC

svc= OneVsRestClassifier(BaggingClassifier(SVC(kernel='linear',
                                               probability=True)))
svc.fit(xTrain,yTrain)

Y_pred_SVM = svc.predict(xTest)
confusion_matrix(yTest,Y_pred_SVM)

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

svm_roc_auc= roc_auc_score(yTest,svc.predict(xTest))
fpr,tpr,thresholds = roc_curve(yTest,svc.predict_proba(xTest)[:,1])

plt.figure()
plt.plot(fpr,tpr,label='SVM (area = %0.2f)' %svm_roc_auc)
plt.plot([0,1],[0,1],'r--') #r means color red and -- means style of line
plt.xlim([0.0,1.0]) #limit of x axis
plt.ylim([0.0,1.0]) #limit of y axis
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC and AUC')
plt.legend(loc="lower right") #location of Legend by default is left top
plt.show()
```



In [28]:

```
##MODEL EVALUATION TABLE
#pip install pTable
from prettytable import PrettyTable
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score

x= PrettyTable()
x.fld_names = ["MODEL1","ACCURACY1","PRECISION1","RECALL1","F1-SCORE1","ACCURACY1"]
x.add_row(["LOGISTIC REGRESSION",
           format(round(accuracy_score(yTest,lg_pred),2),"0.2f"),
           format(round(precision_score(yTest,lg_pred),2),"0.2f"),
           format(round(recall_score(yTest,lg_pred),2),"0.2f"),
           format(round(f1_score(yTest,lg_pred),2),"0.2f"),
           format(round(lg_roc_auc,2),"0.2f")])
print(x)

x.add_row(["RANDOM FOREST",
           format(round(accuracy_score(yTest,Y_prediction),2),"0.2f"),
           format(round(precision_score(yTest,Y_prediction),2),"0.2f"),
           format(round(recall_score(yTest,Y_prediction),2),"0.2f"),
           format(round(f1_score(yTest,Y_prediction),2),"0.2f"),
           format(round(rf_roc_auc,2),"0.2f")])
print(x)
```

```
+---------------------+---------+---------+---------+---------+---------+
|       Field 1       | Field 2 | Field 3 | Field 4 | Field 5 | Field 6 |
+---------------------+---------+---------+---------+---------+---------+
| LOGISTIC REGRESSION |   0.68  |   0.46  |   0.20  |   0.28  |   0.55  |
+---------------------+---------+---------+---------+---------+---------+
+---------------------+---------+---------+---------+---------+---------+
|       Field 1       | Field 2 | Field 3 | Field 4 | Field 5 | Field 6 |
+---------------------+---------+---------+---------+---------+---------+
| LOGISTIC REGRESSION |   0.68  |   0.46  |   0.20  |   0.28  |   0.55  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
+---------------------+---------+---------+---------+---------+---------+
```

In [29]:

```
x.add_row(["RANDOM FOREST",
           format(round(accuracy_score(yTest,Y_prediction),2),"0.2f"),
           format(round(precision_score(yTest,Y_prediction),2),"0.2f"),
           format(round(recall_score(yTest,Y_prediction),2),"0.2f"),
           format(round(f1_score(yTest,Y_prediction),2),"0.2f"),
           format(round(rf_roc_auc,2),"0.2f")])
print(x)
```

```
+---------------------+---------+---------+---------+---------+---------+
|       Field 1       | Field 2 | Field 3 | Field 4 | Field 5 | Field 6 |
+---------------------+---------+---------+---------+---------+---------+
| LOGISTIC REGRESSION |   0.68  |   0.46  |   0.20  |   0.28  |   0.55  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
+---------------------+---------+---------+---------+---------+---------+
```

In [30]:

```
x.add_row(["SVM",
           format(round(accuracy_score(yTest,Y_pred_SVM),2),"0.2f"),
           format(round(precision_score(yTest,Y_pred_SVM),2),"0.2f"),
           format(round(recall_score(yTest,Y_pred_SVM),2),"0.2f"),
           format(round(f1_score(yTest,Y_pred_SVM),2),"0.2f"),
           format(round(svm_roc_auc,2),"0.2f")])
print(x)
```

```
+---------------------+---------+---------+---------+---------+---------+
|       Field 1       | Field 2 | Field 3 | Field 4 | Field 5 | Field 6 |
+---------------------+---------+---------+---------+---------+---------+
| LOGISTIC REGRESSION |   0.68  |   0.46  |   0.20  |   0.28  |   0.55  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|         SVM         |   0.69  |   0.00  |   0.00  |   0.00  |   0.50  |
+---------------------+---------+---------+---------+---------+---------+
```

In [31]:

```
x.add_row(["DECISION TREE",
           format(round(accuracy_score(yTest,Y_Pred_DT),2),"0.2f"),
           format(round(precision_score(yTest,Y_Pred_DT),2),"0.2f"),
           format(round(recall_score(yTest,Y_Pred_DT),2),"0.2f"),
           format(round(f1_score(yTest,Y_Pred_DT),2),"0.2f"),
           format(round(dt_roc_auc,2),"0.2f")])
print(x)
```

```
+---------------------+---------+---------+---------+---------+---------+
|       Field 1       | Field 2 | Field 3 | Field 4 | Field 5 | Field 6 |
+---------------------+---------+---------+---------+---------+---------+
| LOGISTIC REGRESSION |   0.68  |   0.46  |   0.20  |   0.28  |   0.55  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|         SVM         |   0.69  |   0.00  |   0.00  |   0.00  |   0.50  |
|    DECISION TREE    |   0.63  |   0.40  |   0.39  |   0.39  |   0.56  |
+---------------------+---------+---------+---------+---------+---------+
```

In [32]:

```
x.add_row(["KNN",
           format(round(accuracy_score(yTest,Y_pred_knn),2),"0.2f"),
           format(round(precision_score(yTest,Y_pred_knn),2),"0.2f"),
           format(round(recall_score(yTest,Y_pred_knn),2),"0.2f"),
           format(round(f1_score(yTest,Y_pred_knn),2),"0.2f"),
           format(round(knn_roc_auc,2),"0.2f")])
print(x)
```

```
+---------------------+---------+---------+---------+---------+---------+
|       Field 1       | Field 2 | Field 3 | Field 4 | Field 5 | Field 6 |
+---------------------+---------+---------+---------+---------+---------+
| LOGISTIC REGRESSION |   0.68  |   0.46  |   0.20  |   0.28  |   0.55  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|    RANDOM FOREST    |   0.68  |   0.47  |   0.37  |   0.42  |   0.59  |
|         SVM         |   0.69  |   0.00  |   0.00  |   0.00  |   0.50  |
|    DECISION TREE    |   0.63  |   0.40  |   0.39  |   0.39  |   0.56  |
|         KNN         |   0.70  |   0.56  |   0.15  |   0.24  |   0.55  |
+---------------------+---------+---------+---------+---------+---------+
```

In [ ]: