1. import nessasary libraries

In [1]:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

2. import DataSets

In [2]:

```
data=pd.read_csv("claimants.csv")
data.head()
```

Out[2]:

	CASENUM	ATTORNEY	CLMSEX	CLMINSUR	SEATBELT	CLMAGE	LOSS
0	5	0	0.0	1.0	0.0	50.0	34.940
1	3	1	1.0	0.0	0.0	18.0	0.891
2	66	1	0.0	1.0	0.0	5.0	0.330
3	70	0	0.0	1.0	1.0	31.0	0.037
4	96	1	0.0	1.0	0.0	30.0	0.038

3.Data Undestanding

In [3]:

```
data.isnull().sum()
```

Out[3]:

```
CASENUM       0
ATTORNEY      0
CLMSEX       12
CLMINSUR     41
SEATBELT     48
CLMAGE      189
LOSS          0
dtype: int64
```

In [4]:

```
data=data.dropna()
```

In [5]:

```
data=data.drop(['CASENUM'],axis=1)
```

In [6]:

```
data.shape
```

Out[6]:

```
(1096, 6)
```

In [7]:

```
data.dtypes
```

Out[7]:

```
ATTORNEY      int64
CLMSEX      float64
CLMINSUR    float64
SEATBELT    float64
CLMAGE      float64
LOSS        float64
dtype: object
```

Model building

In [8]:

```
x=data.drop(['ATTORNEY'],axis=1)
y=data[['ATTORNEY']]
```

In [9]:

```
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=12,stratify=
```

In [10]:

```
from sklearn.linear_model import LogisticRegression
linear_model=LogisticRegression()
linear_model.fit(x_train,y_train)
```

Out[10]:

```
LogisticRegression()
```

In [29]:

```
linear_model.coef_
```

Out[29]:

```
array([[ 0.54894944,  0.7041359 , -0.37355842,  0.00691333, -0.39317982]])
```

In [31]:

```
from sklearn.tree import DecisionTreeClassifier
dt_model=DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[31]:

```
DecisionTreeClassifier()
```

In [44]:

```
y_train_pred=dt_model.predict(x_train)
```

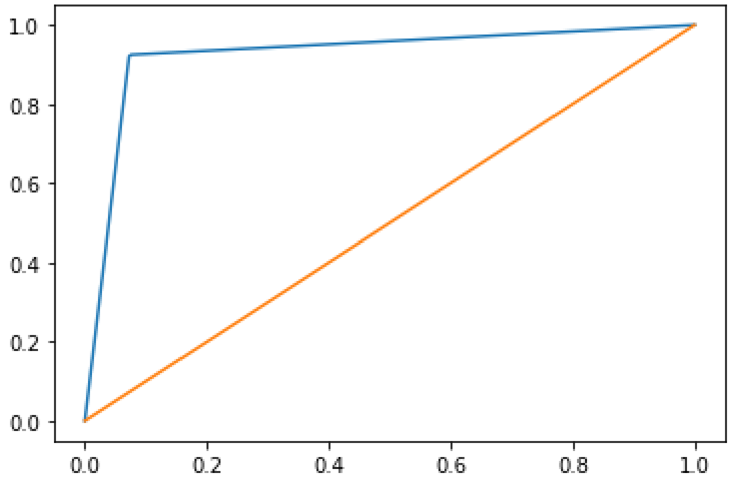
In [40]:

```
from sklearn.metrics import roc_auc_score,roc_auc_score,roc_curve,roc_auc_score
import matplotlib.pyplot as plt
fpr,tpr,thresholds=roc_curve(y,dt_model.predict_proba(x)[:,1])
auc=roc_auc_score(y_train,y_train_pred)
print(auc)
plt.plot(fpr,tpr)
plt.plot([0,1],[0,1])
```

```
0.9975845410628019
```

Out[40]:

```
[<matplotlib.lines.Line2D at 0x23765b5e700>]
```



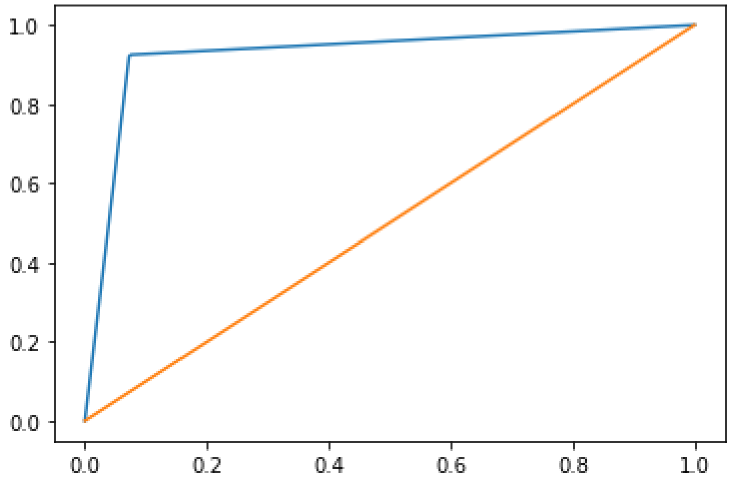
In [59]:

```
from sklearn.metrics import roc_auc_score,roc_curve
fpr,tpr,threshold=roc_curve(y,dt_model.predict_proba(x)[:,1])

plt.plot(fpr,tpr)
plt.plot([0,1],[0,1])
```

Out[59]:

```
[<matplotlib.lines.Line2D at 0x23766d221c0>]
```



In [61]:

```
from  sklearn import datasets
```

In [71]:

```
print(data.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is take
n
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field an
d
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to
a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analys
```

```
is.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transacti
ons
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

In [80]:

```
data=datasets.load_iris()
data_df=pd.DataFrame(data.data,columns=data.feature_names)
data_df['target']=data.target
data_df
```

Out[80]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

3.Data Undestanding

In [84]:

```
data_df.shape
```

Out[84]:

```
(150, 5)
```

In [87]:

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

4.Model Building

In [91]:

```
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=12,shuffle=y
x_train.shape,y_train.shape
```

Out[91]:

```
((876, 5), (876, 1))
```

In [11]:

```
from sklearn.linear_model import LogisticRegression
l_model=LogisticRegression()
l_model.fit(x_train,y_train)
```

Out[11]:

```
LogisticRegression()
```

In [94]:

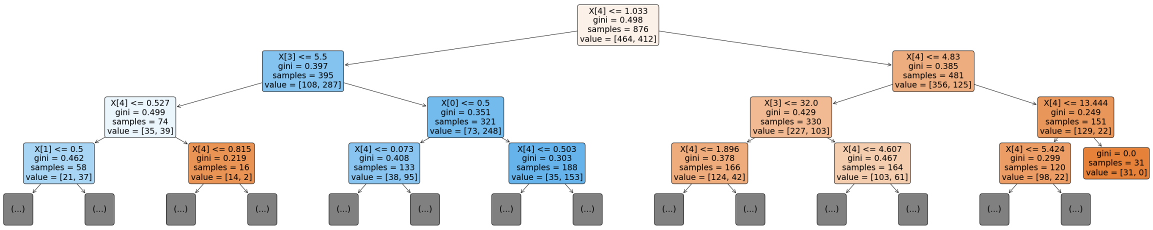
```
from sklearn.tree import DecisionTreeClassifier
dt_model=DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[94]:

```
DecisionTreeClassifier()
```

In [103]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,10))
plot_tree(dt_model,max_depth=3,filled=True,rounded=True)
plt.show()
```



In [106]:

```
y_train_pred=dt_model.predict(x_train)
y_test_pred=dt_model.predict(x_test)
```

Out[106]:

```
array([1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1],
      dtype=int64)
```

3.Model Evalution

In [119]:

```
from sklearn.metrics import accuracy_score,mean_absolute_error,mean_squared_error,confusion
```

In [121]:

```
confusion_matrix(y_test,y_test_pred)
```

Out[121]:

```
array([[64, 50],
       [42, 64]], dtype=int64)
```

In [116]:

```
y_pred_train=l_model.predict(x_train)
y_pred_test=l_model.predict(x_test)
```

In [117]:

```
print(mean_absolute_error(y_train,y_pred_train))
print(mean_squared_error(y_train,y_pred_train))
```

```
0.2888127853881279
0.2888127853881279
```

In [118]:

```
print(mean_absolute_error(y_test,y_pred_test))
print(mean_squared_error(y_test,y_pred_test))
```

```
0.31363636363636366
0.31363636363636366
```