Tic Tac Toe Dataset

Tic Tac Toe has very simple rules.

The goal of a Machine Learning Model for game would keep to implement some kind of pattern that can determine which move is good by regarding the situation on the board.As the possibilities of different games are so limited it is actually possible to solve.But the fun piece is that, I want the model to figure out good moves itself. So we need to make models that can play the game.

Import Modules

First of all, we need to import the required modules.numpy used for making mathematical calculation more accurate pandas used for workinging with file format like csv and seabon and matplotlib will used to plot graphs.

Let's start with importing some modules:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Gathering Data

it's time for first stage of Machine Learning Model which is **Gathering Data**. This stage is very important for predicting models. In this case, the data we collect will be the class and the outcomes of each games

```
data=pd.read_csv('tic-tac-toe.csv')
```

Check Dataset Type

```python
type(data)
```

```
pandas.core.frame.DataFrame
```

```python
data.head()
```

	TL	TM	TR	ML	MM	MR	BL	BM	BR	Class
0	x	x	x	x	o	o	x	o	o	positive
1	x	x	x	x	o	o	o	x	o	positive
2	x	x	x	x	o	o	o	o	x	positive
3	x	x	x	x	o	o	o	b	b	positive
4	x	x	x	x	o	o	b	o	b	positive

Check shape of dataset,column names, Feature column names, Target column name

```python
data.shape
```

```
(958, 10)
```

```python
data.columns
```

```
Index(['TL', 'TM', 'TR', 'ML', 'MM', 'MR', 'BL', 'BM', 'BR', 'Class'], dtype='object')
```

```python
# Extract target column 'passed'
target = print(data.columns[-1])
```

```
Class
```

```python
# Extract Features columns
features = print(data.columns[:-1])
```

```
Index(['TL', 'TM', 'TR', 'ML', 'MM', 'MR', 'BL', 'BM', 'BR'], dtype='object')
```

Now check the most important thing about the dataset by use .info(). By this code we get to know that dataset has 10 columns and all columns are categorical.

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 958 entries, 0 to 957
Data columns (total 10 columns):
TL       958 non-null object
TM       958 non-null object
TR       958 non-null object
ML       958 non-null object
MM       958 non-null object
MR       958 non-null object
BL       958 non-null object
BM       958 non-null object
BR       958 non-null object
Class    958 non-null object
dtypes: object(10)
memory usage: 75.0+ KB
```

It's time to check whether the dataset has Duplicate values or not by using .duplicated() and also check the presence of null values in it both in tabular form and graphical form by using .isnull()

```python
#Check Duplicates
data.duplicated().sum()
```

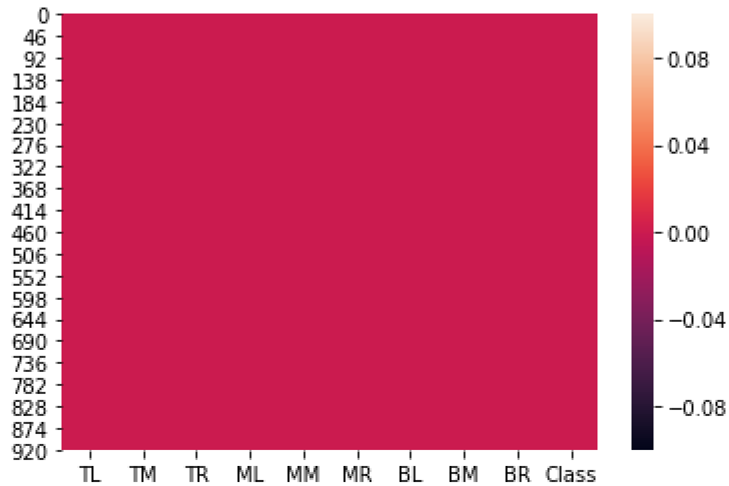
```
0
```

```python
#Check Missing Values
data.isnull().sum()
```

```
TL       0
TM       0
TR       0
ML       0
MM       0
MR       0
BL       0
BM       0
BR       0
Class    0
dtype: int64
```

```python
#Check Missing values in Heatmap
sns.heatmap(data.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bd5979648>
```



As we know the columns in this dataset are categorical, so we need to check their uniqueness

```
data.nunique()
```

```
TL       3
TM       3
TR       3
ML       3
MM       3
MR       3
BL       3
BM       3
BR       3
Class    2
dtype: int64
```

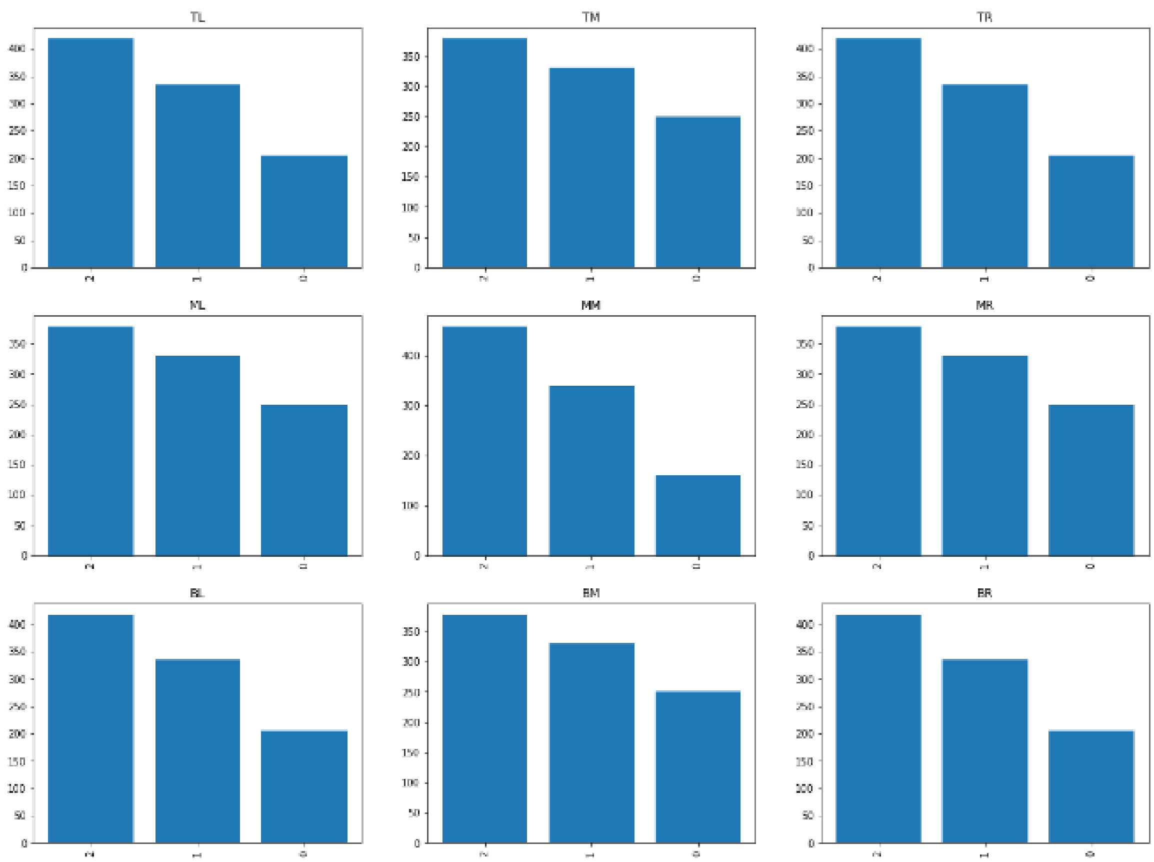
We have to convert those categorical type variables to numbers by Label Encoding

```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

```
for col in data.columns:
    dtype=data[col].dtypes
    if dtype=='object':
        le=LabelEncoder()
        data[col]=le.fit_transform(data[col])
```

Univariate

```
cat_col = ['TL', 'TM', 'TR', 'ML', 'MM', 'MR', 'BL', 'BM','BR']
fig, axs = plt.subplots(3, 3, sharex=False, sharey=False, figsize=(20, 15))

c = 0
for cat_col in cat_col:
    value_counts = data[cat_col].value_counts()

    x = c // 3
    y = c % 3
    x_pos = np.arange(0, len(value_counts))
    axs[x,y].bar(x_pos, value_counts.values, tick_label = value_counts.index)

    axs[x,y].set_title(cat_col)

    for tick in axs[x,y].get_xticklabels():
        tick.set_rotation(90)

    c += 1

plt.show()
```



After Visualization of aour dataset we compute summary of statistics for the Dataframe columns by .describe(). This function gives thecounts, mean, std dev, min, max, 25% percentile, 50% percentile, 75% percentile.

Description

```
data.describe()
```

	TL	TM	TR	ML	MM	MR	BL	BM	BR	Class
count	958.000000	958.000000	958.000000	958.000000	958.000000	958.000000	958.000000	958.000000	958.000000	958.000000
mean	1.222338	1.133612	1.222338	1.133612	1.311065	1.133612	1.222338	1.133612	1.222338	0.653445
std	0.775569	0.798966	0.775569	0.798966	0.740882	0.798966	0.775569	0.798966	0.775569	0.476121
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
75%	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	1.000000
max	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	1.000000

Correlation

By correlation technique that can show the relation between variables by .corr(). By this we can show how strong te relation among variables in both tabular form and graphical form.

```
data.corr()
```

	TL	TM	TR	ML	MM	MR	BL	BM	BR	Class
TL	1.000000	0.002598	-0.061424	0.002598	-0.095030	-0.154229	-0.061424	-0.154229	-0.026680	0.039097
TM	0.002598	1.000000	0.002598	-0.101657	-0.049103	-0.101657	-0.154229	0.050578	-0.154229	-0.084167
TR	-0.061424	0.002598	1.000000	-0.154229	-0.095030	0.002598	-0.026680	-0.154229	-0.061424	0.039097
ML	0.002598	-0.101657	-0.154229	1.000000	-0.049103	0.050578	0.002598	-0.101657	-0.154229	-0.084167
MM	-0.095030	-0.049103	-0.095030	-0.049103	1.000000	-0.049103	-0.095030	-0.049103	-0.095030	0.175583
MR	-0.154229	-0.101657	0.002598	0.050578	-0.049103	1.000000	-0.154229	-0.101657	0.002598	-0.084167
BL	-0.061424	-0.154229	-0.026680	0.002598	-0.095030	-0.154229	1.000000	0.002598	-0.061424	0.039097
BM	-0.154229	0.050578	-0.154229	-0.101657	-0.049103	-0.101657	0.002598	1.000000	0.002598	-0.084167
BR	-0.026680	-0.154229	-0.061424	-0.154229	-0.095030	0.002598	-0.061424	0.002598	1.000000	0.039097
Class	0.039097	-0.084167	0.039097	-0.084167	0.175583	-0.084167	0.039097	-0.084167	0.039097	1.000000

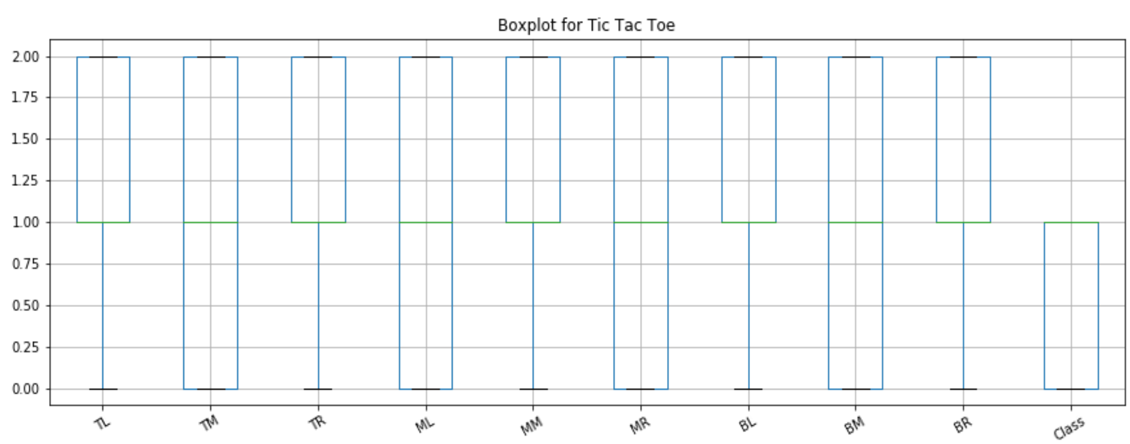
```
#showing the corelation with a heatmap
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(),annot=True,fmt='.2f',vmax=1,vmin=0)
plt.show()
```

Outliers Treatment

Now plot boxplot to display the distribution of data based on five number summary. It can tell us about the presence of Outliers .

```
data.plot(kind='box',grid=True,title="Boxplot for Tic Tac Toe",legend=True,rot=30,figsize=(15,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bd458c148>
```



We need to check for outliers , so take a look on Z-score here. Z-score is the numerical measurement which is used in statistics of a value's relationship to the mean of a group of values, measured in terms of std dev from the mean.

```
#check for Outlier
from scipy.stats import zscore
z_score=abs(zscore(data))
print(data.shape)
data= data.loc[(z_score < 3).all(axis=1)]
print(data.shape)
```

Skewness

We can check Skewwness to get idea on measurement of asymmetry of the probability distribution of real-valued random variable about its mean.

```
data.skew()
```

```
TL       -0.407371
TM       -0.244636
TR       -0.407371
ML       -0.244636
MM       -0.569152
MR       -0.244636
BL       -0.407371
BM       -0.244636
BR       -0.407371
Class    -0.645910
dtype: float64
```

Less than and equal to +/- 0.55 value is an acceptable value for skewwness .But here we get some bigger values,so we have to deal with both bigger values and negetive values in skewness .

```
for col in data.columns:
    if data.skew().loc[col] > 0.55:
        data[col]=np.log1p(data[col])
    if data.skew().loc[col] < -0.55:
        data[col]=np.square(data[col])
```

```
data.skew()
```

```
TL       -0.407371
TM       -0.244636
TR       -0.407371
ML       -0.244636
MM       -0.037478
MR       -0.244636
BL       -0.407371
BM       -0.244636
BR       -0.407371
Class    -0.645910
dtype: float64
```

It's time to split dataset as features columns and target column.

```
x=data.drop("Class",axis=1)
y=data["Class"]
```

```
y=np.array(y)
y=y.reshape(-1,1)
y.shape
```

```
(958, 1)
```

To transform the data in such a manner that it has mean as 0 and std dev ias 1 we apply the Standard Scalling technique. By this we can arrenge the data in a standard normal distribution.

```python
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

x=sc.fit_transform(x)
x
```

```
array([[ 1.00322257,  1.08495342,  1.00322257, ...,  1.00322257,
        -0.16731812, -0.28682739],
       [ 1.00322257,  1.08495342,  1.00322257, ..., -0.28682739,
         1.08495342, -0.28682739],
       [ 1.00322257,  1.08495342,  1.00322257, ..., -0.28682739,
        -0.16731812,  1.00322257],
       ...,
       [-0.28682739,  1.08495342, -0.28682739, ...,  1.00322257,
        -0.16731812,  1.00322257],
       [-0.28682739,  1.08495342, -0.28682739, ...,  1.00322257,
        -0.16731812,  1.00322257],
       [-0.28682739, -0.16731812,  1.00322257, ..., -0.28682739,
         1.08495342,  1.00322257]])
```

Now, we have to split the dataset into training set and testing set.This step is important to make Machine Learning model better.The both train set and test set must be similar, usually same variables.

```python
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict,GridSearchCV
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,f1_score
from sklearn.metrics import roc_curve,roc_auc_score,auc
```

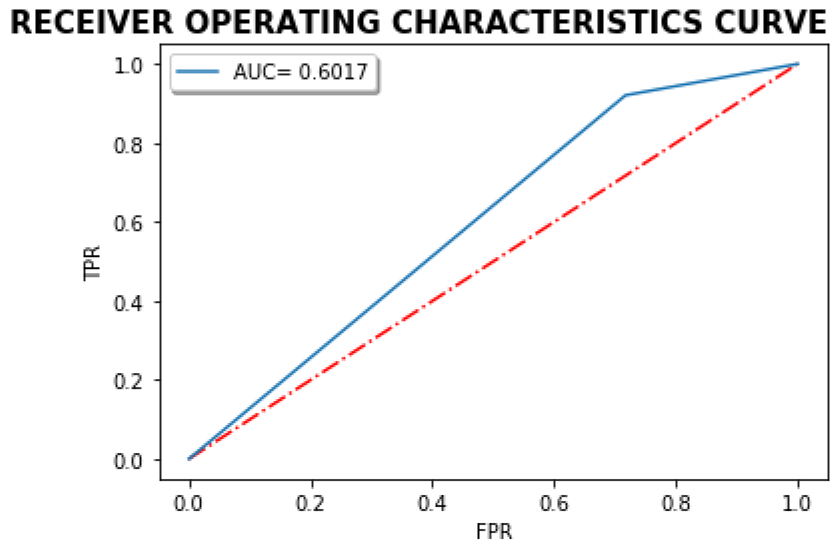
```python
def rst(mod,x,y):
    max_r=0
    for rn_state in range (25,150):
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=rn_state)
        mod.fit(x_train,y_train)
        pred=mod.predict(x_test)
        acs=accuracy_score(pred,y_test)
        f1=f1_score(y_test,pred)
        cnf=confusion_matrix(y_test,pred)
        clr=classification_report(y_test,pred)

        fpr,tpr,thresholds=roc_curve(y_test,pred)
        roc_auc=auc(fpr,tpr)

        if acs > max_r:
            max_r=acs
            random_state=rn_state
    print("random_state for mod"," is ",random_state,"which gives accuracy score of: ",max_r)
    #print('f1_score: ', f1)
    print('confusion matrix: ',cnf)
    print('classification report: ',clr)
    print("fpr: ",fpr)
    print("tpr: ",tpr)
    print("thresholds: ",thresholds)
    print("roc_auc: ",roc_auc)
    print()
    print('prediction: ',pred)
```

```
    plt.plot([0,1],[0,1],color='red',linestyle="dashdot")
    plt.plot(fpr,tpr,label="AUC= %0.4f" % roc_auc)
    plt.legend(loc='best',fontsize='medium',shadow=True)
    plt.xlabel("FPR")
    plt.ylabel('TPR')
    plt.title('RECEIVER OPERATING CHARACTERISTICS CURVE',size=15,weight='bold',loc='right')
    plt.show()

    return random_state
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver="lbfgs")
lr_g=rst(lr,x,y)
```

```
random_state for mod  is  58 which gives accuracy score of:  0.7673611111111112
confusion matrix:  [[ 28  71]
 [ 15 174]]
classification report:                precision    recall  f1-score   support

           0       0.65      0.28      0.39        99
           1       0.71      0.92      0.80       189

    accuracy                           0.70       288
   macro avg       0.68      0.60      0.60       288
weighted avg       0.69      0.70      0.66       288

fpr:  [0.         0.71717172 1.        ]
tpr:  [0.         0.92063492 1.        ]
thresholds:  [2 1 0]
roc_auc:  0.6017316017316017


 prediction:  [1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0
  1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0
  1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
  0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1
  1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
  0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1
  1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1]
```




```
#Cross Validation
val_lr=cross_val_score(lr,x,y,scoring='accuracy',cv=10).mean()
val_lr
```

```
0.6339050913365888
```

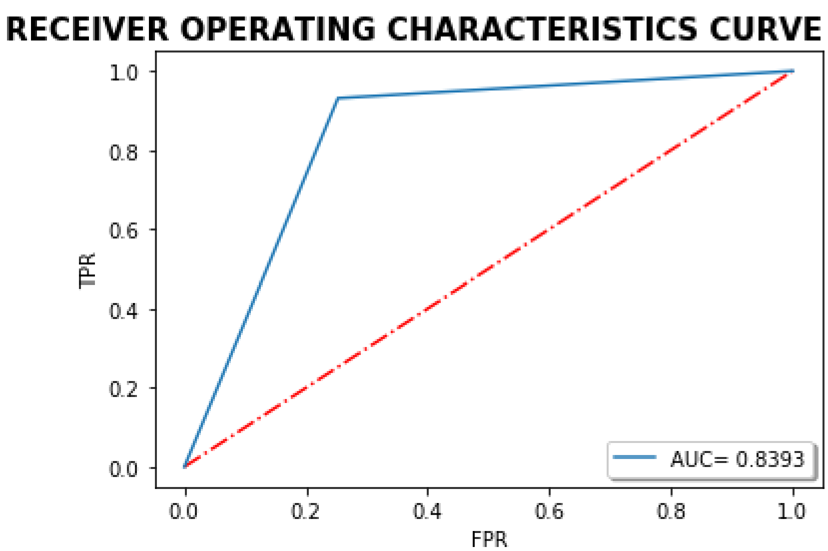
DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt_param={'criterion':['gini','entropy'],'max_depth':range(3,10)}
dt_g=GridSearchCV(dt,dt_param,cv=15)
dt_g.fit(x,y)
dt_g.best_params_
```

```
{'criterion': 'gini', 'max_depth': 9}
```

```
dt=DecisionTreeClassifier(criterion='gini',max_depth=9)
dtc=rst(dt,x,y)
```

```
random_state for mod  is  137 which gives accuracy score of:  0.9375
confusion matrix:  [[ 74  25]
 [ 13 176]]
classification report:                precision    recall  f1-score   support

           0       0.85      0.75      0.80        99
           1       0.88      0.93      0.90       189

    accuracy                           0.87       288
   macro avg       0.86      0.84      0.85       288
weighted avg       0.87      0.87      0.87       288

fpr:  [0.         0.25252525 1.        ]
tpr:  [0.         0.93121693 1.        ]
thresholds:  [2 1 0]
roc_auc:  0.8393458393458392

prediction:  [0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1
 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1
 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 1 0 1 0 1 1
 0 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1
 0 0 0 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1
 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 1 1
 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 0 1
 1 1 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1]
```




```
#Cross Validation
val_dt=cross_val_score(dt,x,y,scoring='accuracy',cv=15).mean()
val_dt
```

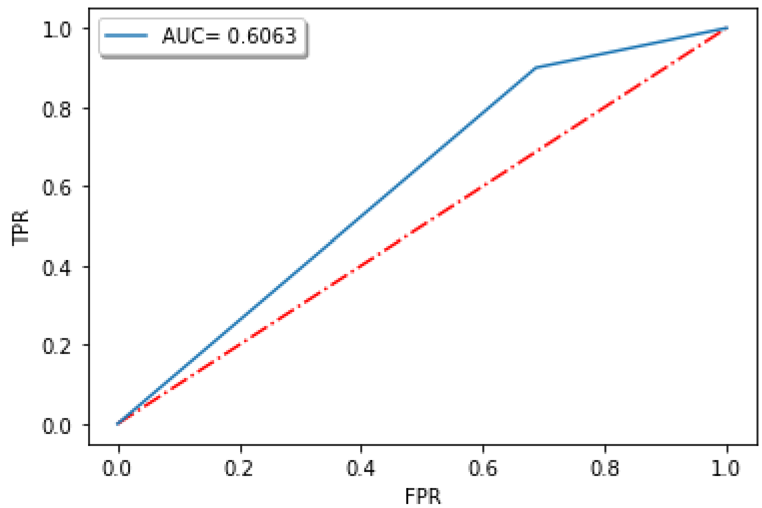
```
0.7696090252340253
```

GaussianNB

```python
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
gnb=rst(nb,x,y)
```

```
random_state for mod  is  94 which gives accuracy score of:  0.7673611111111112
confusion matrix:  [[ 31  68]
 [ 19 170]]
classification report:               precision    recall  f1-score   support

           0       0.62      0.31      0.42        99
           1       0.71      0.90      0.80       189

    accuracy                           0.70       288
   macro avg       0.67      0.61      0.61       288
weighted avg       0.68      0.70      0.67       288

fpr:  [0.         0.68686869 1.        ]
tpr:  [0.        0.8994709 1.       ]
thresholds:  [2 1 0]
roc_auc:  0.6063011063011063

prediction:  [1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1
 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0
 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1
 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1
 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1
 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1]
```

RECEIVER OPERATING CHARACTERISTICS CURVE




```python
#Cross Validation
val_nb=cross_val_score(nb,x,y,scoring='accuracy',cv=5).mean()
val_nb
```

```
0.6370419130745372
```

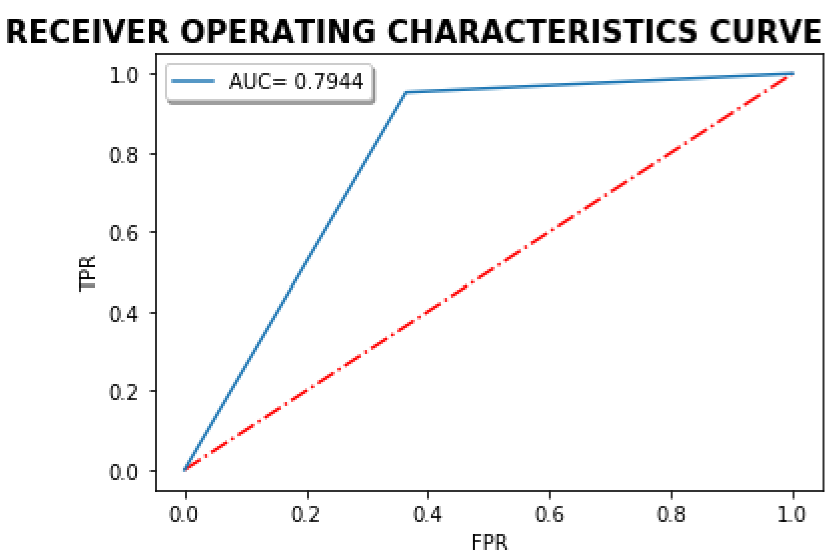
KNN

```python
from sklearn.neighbors import KNeighborsClassifier
kn=KNeighborsClassifier()
kn_param={'n_neighbors':range(1,10),'leaf_size':range(20,70)}
kn_g=GridSearchCV(kn,kn_param,cv=15)
kn_g.fit(x,y)
kn_g.best_params_
```

```
{'leaf_size': 20, 'n_neighbors': 5}
```

```python
kn=KNeighborsClassifier(n_neighbors=5,leaf_size=20)
knn=rst(kn,x,y)
```

```
random_state for mod  is  137 which gives accuracy score of:  0.8923611111111112
confusion matrix:  [[ 63  36]
 [  9 180]]
classification report:                precision    recall  f1-score   support

           0       0.88      0.64      0.74        99
           1       0.83      0.95      0.89       189

    accuracy                           0.84       288
   macro avg       0.85      0.79      0.81       288
weighted avg       0.85      0.84      0.84       288

fpr:  [0.         0.36363636 1.        ]
tpr:  [0.         0.95238095 1.        ]
thresholds:  [2 1 0]
roc_auc:  0.7943722943722944

prediction:  [0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1
 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1
 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0
 0 1 1 1 1 1 1 0 1 0 1 0 0 1 1 1 0 0 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1
 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1
 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1
 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1
 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 0 1 1 0 1 0 1 1 1]
```



```
#Cross Validation
val_kn=cross_val_score(kn,x,y,scoring='accuracy',cv=10).mean()
val_kn
```

```
0.8291542322300597
```

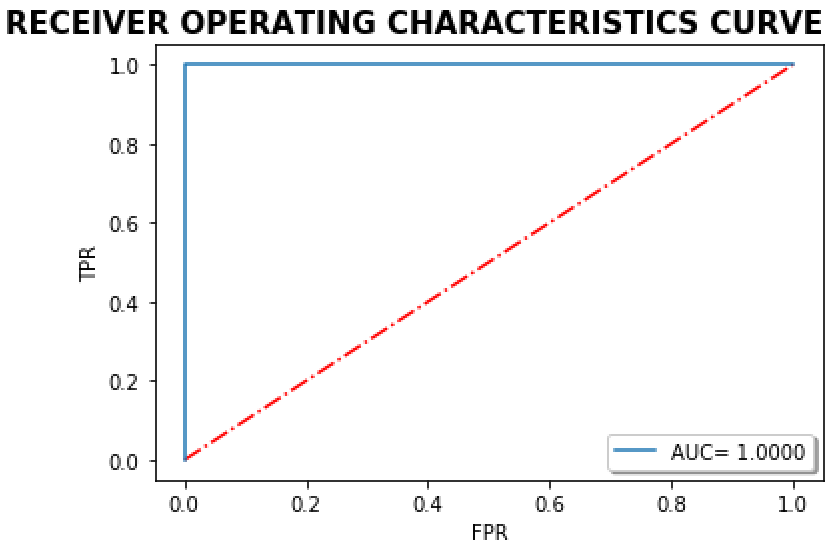
Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=85)

gb_param= {"learning_rate": [0.0001,0.025,0.1,1.0], "n_estimators":[250,500,750,1000]}

gb_g= GridSearchCV(gb,gb_param,cv=15)
gb_g.fit(x_train,y_train)
print("best parameters:", gb_g.best_params_)
print("\n best score:",gb_g.best_score_)
```

```
best parameters: {'learning_rate': 0.1, 'n_estimators': 500}

 best score: 1.0
```

```
gb=GradientBoostingClassifier(learning_rate=0.1, n_estimators=500)
gbc=rst(gb,x,y)
```

```
random_state for mod  is  25 which gives accuracy score of:  1.0
confusion matrix:  [[ 99   0]
 [  0 189]]
classification report:               precision    recall  f1-score   support

           0       1.00      1.00      1.00        99
           1       1.00      1.00      1.00       189

    accuracy                           1.00       288
   macro avg       1.00      1.00      1.00       288
weighted avg       1.00      1.00      1.00       288

fpr:  [0. 0. 1.]
tpr:  [0. 1. 1.]
thresholds:  [2 1 0]
roc_auc:  1.0

prediction:  [0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1
 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1
 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0
 0 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 1 0 1
 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 0
 0 0 0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1
 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1
 1 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1]
```




```
#Cross Validation
val_gb=cross_val_score(gb,x,y,scoring='accuracy',cv=10).mean()
val_gb
```

```
0.9608247422680412
```

Conclusion

Here we make different Machine Learning Models and get know that most of the models are Overfiited. So by using Grdient Boosting Model we are trying to get a best fit model and here is the result, we got it.

Save model in Joblib

Now it's time to save the best model for using later on by Pickle.

```
import joblib
joblib.dump(gb,"GB_TicTacToe.pkl")
```