Binary Joint Use Case (Single DataFrameCase)

In this vignette a use case of the Binary Channel Entropy Triangle is presented. We are going to evaluate different multiclass-classification scenarios in order to analyze the data. The main functionalities for the classification of the database will be extracted from: https://www.geeksforgeeks.org/multiclass-classification-using-scikit-learn/ (https://www.geeksforgeeks.org/multiclass-classification-using-scikit-learn/)

Importing Libraries

As the functions for the entopies measures are stored in other domain, first we will need to access those modules with the functions and the import all the necessary functions

In [1]:

```
# Bring your packages onto the path
import sys,os
sys.path.append(os.path.abspath(os.path.join('..'))) #'entropytriangle main directory
```

In [2]:

```
from entropytriangle import * #importing all modules necessary for the plotting
```

Download the databases

In [3]:

```
#df = pd.read_csv('Arthritis.csv',delimiter=',',index_col='Unnamed: 0')
df = pd.read_csv('Breast_data.csv',delimiter=',',index_col='Unnamed: 0').drop(['Sample code number'],axis = 1).replace('?',np.nan) # in this DB the missing values are represented as '?'
#df = pd.read_csv('Glass.csv',delimiter=',')
#df = pd.read_csv('Ionosphere.csv',delimiter=',')
#df = pd.read_csv('Iris.csv',delimiter=',',index_col='Id')
#df = pd.read_csv('Wine.csv',delimiter=',').drop(['Wine'],axis = 1)
```

In [4]:

```
df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 1 to 699
Data columns (total 10 columns):
Clump Thickness                699 non-null int64
Uniformity of Cell Size        699 non-null int64
Uniformity of Cell Shape       699 non-null int64
Marginal Adhesion              699 non-null int64
Single Epithelial Cell Size    699 non-null int64
Bare Nuclei                    683 non-null float64
Bland Chromatin                699 non-null int64
Normal Nucleoli                699 non-null int64
Mitoses                        699 non-null int64
Class                          699 non-null object
dtypes: float64(1), int64(8), object(1)
memory usage: 60.1+ KB
```

In [5]:

```
df = df.fillna(0)
df.head(5)
```

Out[5]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitose
1	5	1	1	1	2	1.0	3	1	
2	5	4	4	5	7	10.0	3	2	
3	3	1	1	1	2	2.0	3	1	
4	6	8	8	1	3	4.0	3	7	
5	4	1	1	3	2	1.0	3	1	

Prepare the data for the classification (Features - Classes)

We are going to load the train_test_split that will allow us to separe automatically the data in a train/test sets. Additionally, we are going to import the contingency matrix that will allow us to calculate the joint entropy matrix of the classifier

In [6]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

Separating the data farmes features and classes

In [7]:

```
X = df[df.columns[df.columns != 'Class']]
y = df['Class']
```

We are now to define some classificators for evaluating their performance with the BreastCancer database

In [8]:

```
# dividing X, y into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

KNN

KNN - Classifier (Don´t run the code if you want to implement other classifier)

Downloading the sklearn Knn classifier and fitting it into our data

In [9]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
```

Out[9]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

Once we have design our classifier, we are going to evaluate the accuracy

In [10]:

```
print(knn.score(X_test, y_test))
```

```
0.9771428571428571
```

Finally, we will compute the confusion matrix of the classified data

In [11]:

```
knn_predictions = knn.predict(X_test)
cm = confusion_matrix(y_test, knn_predictions)
cm
```

Out[11]:

```
array([[110,   2],
       [  2,  61]])
```

KNN - Channel Bivariate Entropy Triangle Plotting

The last step will be calculating the entropic measures for the contingency matrix and plot the entropy triangle. The coordinates will be calculated multiplying the normalized values needed by the scale used for plotting the triangle, and will appear behind the triangle plot for comparission

In [12]:

```
edf = jentropies_binary(cm)
#edf1 = jentropies(pd.DataFrame(y_test),pd.DataFrame(knn_predictions))
```

In [13]:

```
edf
#edf1
```

Out[13]:

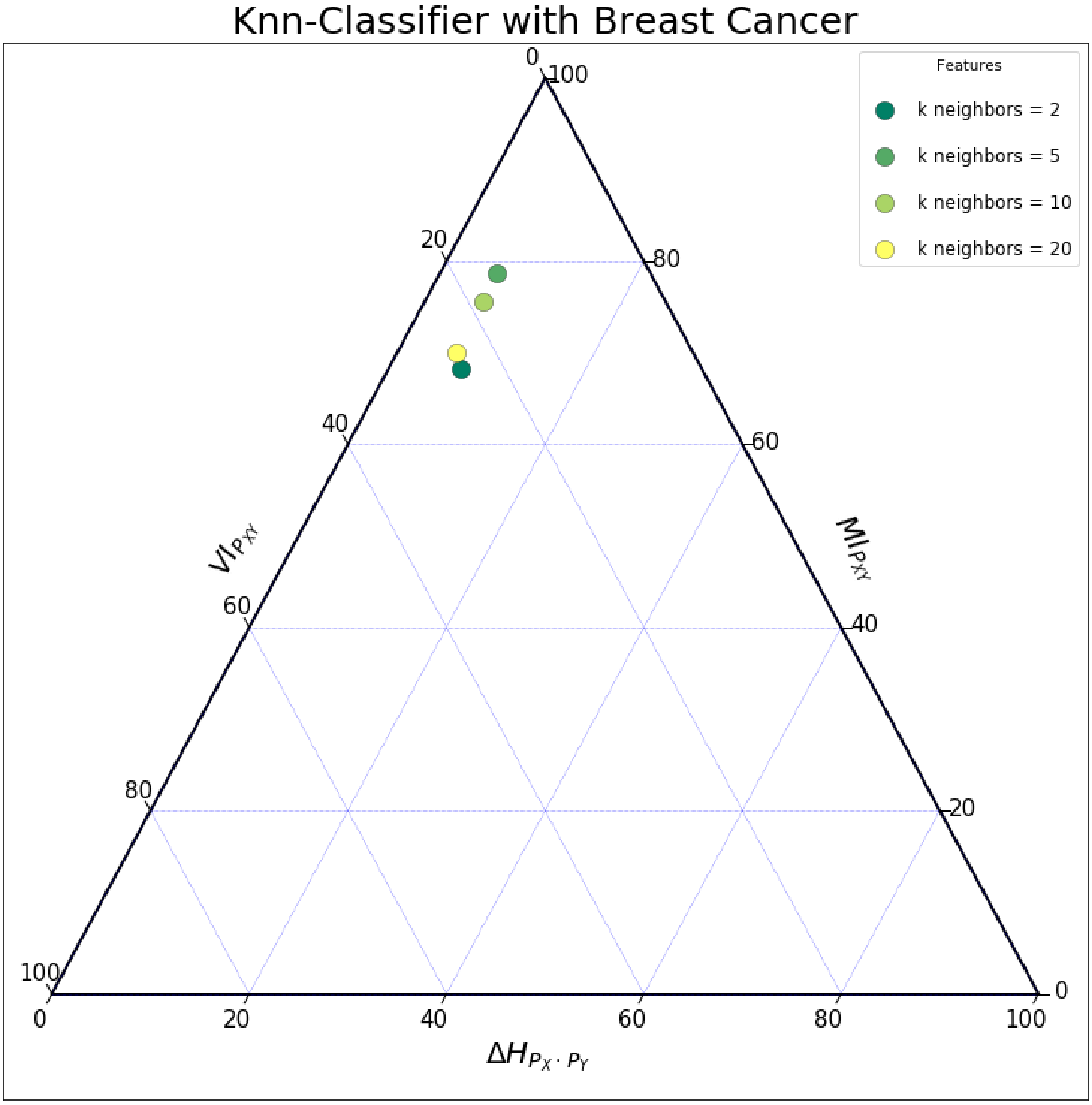
	H_U2	H_P2	DeltaH_P2	M_P2	VI_P2
Type					
X	1.0	0.942683	0.057317	0.786867	0.155816
Y	1.0	0.942683	0.057317	0.786867	0.155816
XY	2.0	1.885366	0.114634	1.573734	0.311632

In [16]:

```
l = [2,5,10,20]
names_l = list(str('k neighbors = '+ str(l[i])) for i in range(len(l)))
lis = list()
for i in range(len(l)):
    knn = KNeighborsClassifier(n_neighbors = l[i])
    knn.fit(X_train, y_train)
    knn_predictions = knn.predict(X_test)
    cm = confusion_matrix(y_test, knn_predictions)
    edf = jentropies_binary(cm)
    lis.append(edf.iloc[[2]])
```

In [17]:

```
entriangle_list(lis,names=names_l,s_mk=150, gridl = 20, pltscale=13 ,fonts = 20,
 ticks_size= 15,chart_title="Knn-Classifier with Breast Cancer")
```


Knn-Classifier with Breast Cancer
Features
● k neighbors = 2
● k neighbors = 5
● k neighbors = 10
● k neighbors = 20
$VI_{P_{XY}}$
$MI_{P_{XY}}$
$\Delta H_{P_X \cdot P_Y}$