In [3]:

```
import numpy as np
import pandas as pd


from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC


from sklearn import datasets, metrics
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

%matplotlib inline
```

Abrir base dados

In [4]:

```
dataset = pd.read_csv('Bases/QuantizationError_teste.csv', encoding='utf
-8')
```

In [47]:

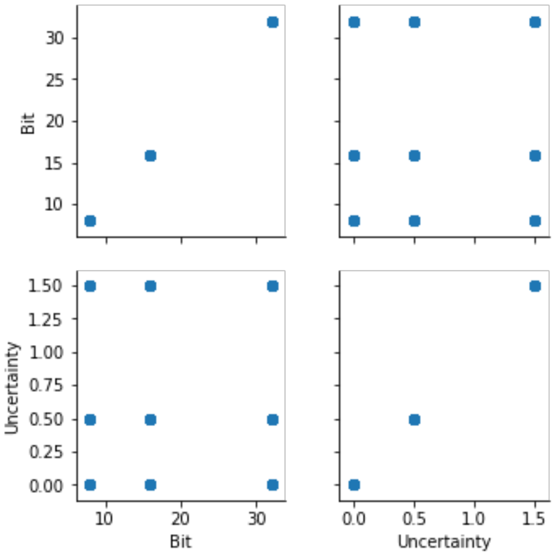
```
dataset.head()
```

Out[47]:

	Digital System Name (original name)	Realization	Bit	Implementation	Uncertainty	Class
0	cruise	<6.2>	8	DFI	0.0	NaN
1	cruise	<6.2>	8	DFI	0.5	NaN
2	cruise	<6.2>	8	DFI	1.5	NaN
3	cruise	<6.2>	8	DFII	0.0	NaN
4	cruise	<6.2>	8	DFII	0.5	NaN

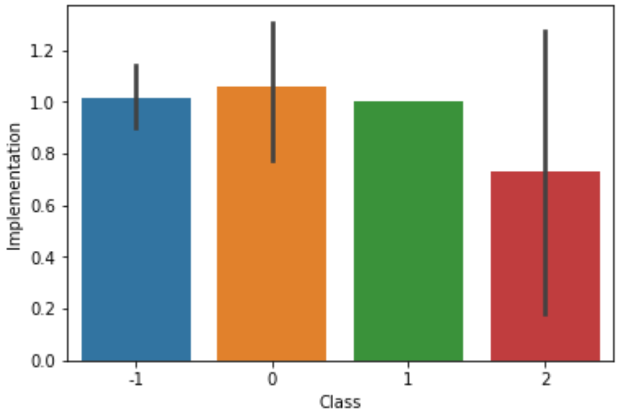
```
In [5]: g = sns.PairGrid(dataset)
        g.map(plt.scatter)
```

```
Out[5]: <seaborn.axisgrid.PairGrid at 0x7faeb84293c8>
```



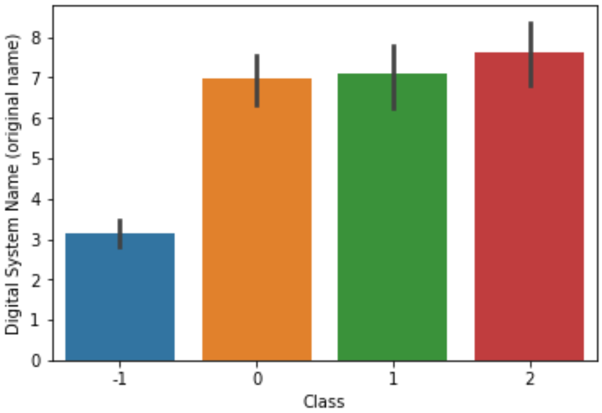
```
In [31]: sns.barplot(x='Class',y='Implementation',data=dataset)
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7faeb60cf4a8>
```



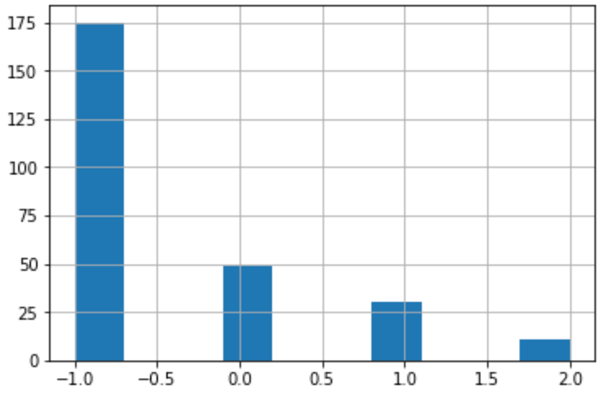
```
In [32]: sns.barplot(x='Class',y='Digital System Name (original name)',data=dataset)

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7faeb630c128>
```



```
In [33]: dataset['Class'].hist()

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7faeb5b73cf8>
```



Converter para numeros

```
In [10]: dataset['Digital System Name (original name)'] = dataset['Digital System Name (original name)'].astype('category').cat.codes
         dataset['Realization'] = dataset['Realization'].astype('category').cat.codes
         dataset['Bit'] = dataset['Bit'].astype('category').cat.codes
         dataset['Implementation'] = dataset['Implementation'].astype('category').cat.codes
         dataset['Uncertainty'] = dataset['Uncertainty'].astype('category').cat.codes
         dataset['Class'] = dataset['Class'].astype('category').cat.codes
```

Converter em um problema supervisionado

```
In [11]: X = dataset.drop(['Class'],axis=1)
         y = dataset['Class']
```

Separar em train e test

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                             test_size=0.30,
                                                             random_state=123,
                                                             stratify=y)
```

SVM

```
In [13]: # Create a classifier clf: a support vector classifier
         clf_SVM = SVC(gamma=0.001, C=1.,kernel='poly')
         clf_SVM.fit(X_train, y_train)
         clf_SVM.predict(X_test);
         y_pred_SVC = clf_SVM.predict(X_test)
```

Avaliação

```
In [14]: #Apenas acurácia
         acc_SVC = metrics.accuracy_score(y_test, y_pred_SVC)
         print("Accuracy: \n%s" % acc_SVC)
```

```
Accuracy:
0.6625
```

```
In [15]: # Relatório de avaliação com outras métricas e matriz de confusão
         print("Classification report for classifier %s:\n%s\n"
               % (clf_SVM, metrics.classification_report(y_test, y_pred_SVC)))
         print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred_
         SVC))
```

```
Classification report for classifier SVC(C=1.0, cache_size=200, class_wei
ght=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False):
             precision    recall  f1-score   support

         -1       0.66      1.00      0.80        53
          0       0.00      0.00      0.00        15
          1       0.00      0.00      0.00         9
          2       0.00      0.00      0.00         3

avg / total       0.44      0.66      0.53        80


Confusion matrix:
[[53  0  0  0]
 [15  0  0  0]
 [ 9  0  0  0]
 [ 3  0  0  0]]
```

```
/home/maria/tensorflow/lib/python3.5/site-packages/sklearn/metrics/classi
fication.py:1135: UndefinedMetricWarning: Precision and F-score are ill-d
efined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Testando kNN

```
In [16]: # Create a classifier clf: a support vector classifier
         clf_kNN = KNeighborsClassifier(n_neighbors=8)
```

```
In [17]: # We learn the digits on training dataset
         nbrs=clf_kNN.fit(X_train, y_train)
```

```
In [18]: y_pred_kNN = nbrs.predict(X_test)
```

Avaliação

```
In [19]: #Apenas acurácia
         acc_kNN = metrics.accuracy_score(y_test, y_pred_kNN)
         print("Accuracy: \n%s" % acc_kNN)
```

```
Accuracy:
0.75
```

```
In [20]: # Relatório de avaliação com outras métricas e matriz de confusão
         print("Classification report for classifier %s:\n%s\n"
               % (clf_kNN, metrics.classification_report(y_test, y_pred_kNN)))
         print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred_
         kNN))
```

```
Classification report for classifier KNeighborsClassifier(algorithm='auto
', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=8, p=2,
           weights='uniform'):
             precision    recall  f1-score   support

         -1       0.87      0.98      0.92        53
          0       0.60      0.40      0.48        15
          1       0.12      0.11      0.12         9
          2       0.50      0.33      0.40         3

avg / total       0.72      0.75      0.73        80


Confusion matrix:
[[52  0  1  0]
 [ 5  6  4  0]
 [ 3  4  1  1]
 [ 0  0  2  1]]
```

DecisionTree

```
In [21]: # Create a classifier clf: a support vector classifier
         clf_DT = DecisionTreeClassifier(max_depth=20)
```

```
In [22]: # We learn the digits on training dataset
         clf_DT=clf_DT.fit(X_train, y_train)
```

```
In [23]: y_pred_DT = clf_DT.predict(X_test)
```

Avaliação

```
In [24]: #Apenas acurácia
         acc_DT = metrics.accuracy_score(y_test, y_pred_DT)
         print("Accuracy: \n%s" % acc_DT)
```

```
Accuracy:
0.925
```

```
In [25]: # Relatório de avaliação com outras métricas e matriz de confusão
         print("Classification report for classifier %s:\n%s\n"
               % (clf_DT, metrics.classification_report(y_test, y_pred_DT)))
         print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred_
         DT))
```

```
Classification report for classifier DecisionTreeClassifier(class_weight=
None, criterion='gini', max_depth=20,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best'):
             precision    recall  f1-score   support

         -1       1.00      1.00      1.00        53
          0       0.92      0.80      0.86        15
          1       0.64      0.78      0.70         9
          2       0.67      0.67      0.67         3

avg / total       0.93      0.93      0.93        80


Confusion matrix:
[[53  0  0  0]
 [ 0 12  3  0]
 [ 0  1  7  1]
 [ 0  0  1  2]]
```

Random Forest

```
In [26]: # Create a classifier clf: a support vector classifier
         clf_rf = RandomForestClassifier(n_estimators=15,random_state=1)
```

```
In [27]: # We learn the digits on training dataset
         clf_rf=clf_rf.fit(X_train, y_train)
```

```
In [28]: y_pred_rf = clf_rf.predict(X_test)
```

Avaliação

```
In [29]: #Apenas acurácia
         acc_rf = metrics.accuracy_score(y_test, y_pred_rf)
         print("Accuracy: \n%s" % acc_rf)
```

```
Accuracy:
0.875
```

In [30]:

```
# Relatório de avaliação com outras métricas e matriz de confusão
print("Classification report for classifier %s:\n%s\n"
      % (clf_rf, metrics.classification_report(y_test, y_pred_rf)))
print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred_
rf))
```

```
Classification report for classifier RandomForestClassifier(bootstrap=Tru
e, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=15, n_jobs=1,
            oob_score=False, random_state=1, verbose=0, warm_start=False)
:
             precision    recall  f1-score   support

         -1       0.94      0.96      0.95        53
          0       0.83      0.67      0.74        15
          1       0.64      0.78      0.70         9
          2       0.67      0.67      0.67         3

avg / total       0.88      0.88      0.87        80


Confusion matrix:
[[51  1  1  0]
 [ 2 10  3  0]
 [ 1  0  7  1]
 [ 0  1  0  2]]
```