```
!pip install ExKMC
!pip install adjustText
```

```
Requirement already satisfied: ExKMC in /usr/local/lib/python3.7/dist-packages (0.0.3)
Requirement already satisfied: adjustText in /usr/local/lib/python3.7/dist-packages (0.7
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from adj
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/li
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycle
```

```
from ExKMC.Tree import Tree
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from adjustText import adjust_text

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import tqdm
# Create dataset
# n = 100
# d = 10
# k = 3
# X, _ = make_blobs(n, d, k, cluster_std=3.0)

# # Initialize tree with up to 6 leaves, predicting 3 clusters
# tree = Tree(k=k, max_leaves=2*k)

# # Construct the tree, and return cluster labels
# prediction = tree.fit_predict(X)

# # Tree plot saved to filename
# tree.plot('filename')
```

```
data = pd.read_excel('variablesClasificacionFRE.xlsx', na_values='-')

data['Departamento.1'] = data['Departamento.1'].str.title()
```

```
data.head(3)
```

	Departamento	Departamento.1	Presupuesto_2020	Presupuesto_Sa
0	23	Amazonas	195000000000	3.89

```
columns = ['Presupuesto_2020', 'Presupuesto_Salud_2020',
             'No_Inscritos', 'Prop_productosCD_2021',
             'Cumplimiento_A1_2020-2021-06', 'Cumplimiento_A2_2020-2021-06',
             'PropPortafolio', 'T_adq_Recet', 'Departamento.1']

X = data.loc[:, columns]
X.set_index('Departamento.1', inplace=True)
X.dropna(inplace=True)
X.head(3)
```

Departamento.1	Presupuesto_2020	Presupuesto_Salud_2020	No_Insc
Amazonas	195000000000	3.892668e+10	
Antioquia	2714000000000	9.783884e+11	

```
scaler = StandardScaler()
scaler.fit(X)
X_train = scaler.transform(X)
```

```
np.shape(X_train)
```

```
(31, 8)
```

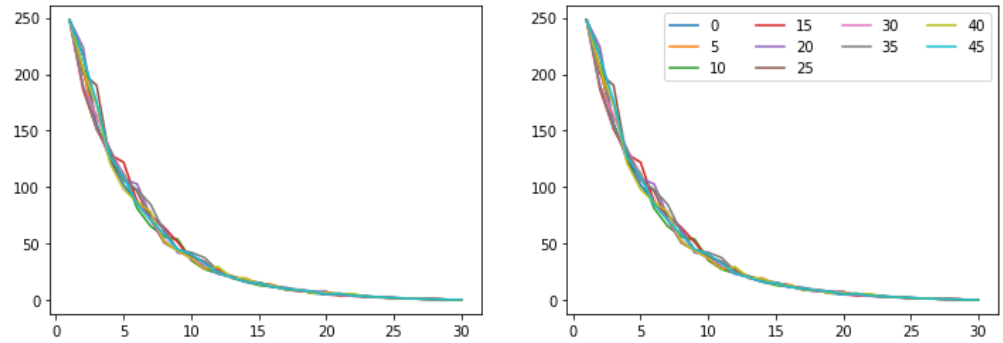
```
## Evaluación de número k
k_list = np.arange(1, 31)
leaves_list = np.arange(1, 50)

lv_list1 = list()
lv_list2 = list()

for lv in tqdm.tqdm(leaves_list):
  score_list1 = list()
  score_list2 = list()

  for i in k_list:
    tree = Tree(k = i, max_leaves = 2 * i)
    prediction = tree.fit_predict(X_train)
    score_list1.append(tree.score(X_train))
    score_list2.append(tree.surrogate_score(X_train))
```

```
  lv_list1.append(score_list1)
  lv_list2.append(score_list2)
```

```
    100%|██████████| 49/49 [00:25<00:00,  1.94it/s]
```

```
fig, ax = plt.subplots(1,2, figsize = (12, 4))

for j, _ in enumerate(leaves_list):
  if j % 5 == 0:
    ax[0].plot(k_list, lv_list1[j], label = j)
for j, _ in enumerate(leaves_list):
  if j % 5 == 0:
    ax[1].plot(k_list, lv_list2[j], label = j)

ax[1].legend(ncol = 4)
```

```
    <matplotlib.legend.Legend at 0x7fce4bc8bc10>
```




```
k = 3
tree = Tree(k=k, max_leaves=2*k)
# Construct the tree, and return cluster labels
prediction = tree.fit_predict(X_train)
```

```
prediction
```

```
    array([1., 0., 1., 0., 2., 1., 1., 1., 1., 1., 0., 1., 2., 2., 1., 1., 1.,
           1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1.])
```

```
tree.plot('filename', feature_names=columns)
```

```
tree.score(X_train)
```

```
    166.6255692175232
```

```
k = 8
tree = Tree(k=k, max_leaves=2*k)
```

```
# Construct the tree, and return cluster labels
prediction = tree.fit_predict(X_train)
prediction
```

```
array([4., 3., 6., 1., 5., 6., 1., 6., 0., 6., 1., 1., 5., 7., 6., 4., 6.,
       0., 0., 6., 6., 6., 6., 1., 2., 1., 6., 1., 1., 2., 2.])
```

```
tree.plot('graficos_k8', feature_names=columns)
```

```
X.columns
```

```
Index(['Presupuesto_2020', 'Presupuesto_Salud_2020', 'No_Inscritos',
       'Prop_productosCD_2021', 'Cumplimiento_A1_2020-2021-06',
       'Cumplimiento_A2_2020-2021-06', 'PropPortafolio', 'T_adq_Recet'],
      dtype='object')
```

```
# X

umb_data = [[-1.194, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN],
            [+2.065, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN],
            [np.NaN, np.NaN, -0.741, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN],
            [np.NaN, np.NaN, -0.727, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN],
            [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, -0.280],
            [np.NaN, np.NaN, np.NaN, -1.487, np.NaN, np.NaN, np.NaN, np.NaN],
            [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 0.548, np.NaN],
            [np.NaN, np.NaN, np.NaN, np.NaN, -3.383, np.NaN, np.NaN, np.NaN],
            [np.NaN, -0.237, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN]]

pd.DataFrame(
    np.transpose(scaler.inverse_transform(umb_data)),
    columns = columns
)
```

	Presupuesto_2020	Presupuesto_Salud_2020	No_Inscritos	Prop_p
0	1.021243e+11	2.005818e+12	NaN	
1	NaN	NaN	NaN	
2	NaN	NaN	3.033913	
3	NaN	NaN	NaN	
4	NaN	NaN	NaN	
5	NaN	NaN	NaN	

```
from sklearn.decomposition import PCA
```

```
pca = PCA(random_state=2021)
pca.fit(X_train)
print(pca.explained_variance_ratio_)
```

```
[0.3919343  0.17741867 0.1231828  0.11761585 0.09167238 0.0736223
 0.0168361  0.00771761]
```

```
[i/sum(pca.explained_variance_ratio_) for i in pca.explained_variance_ratio_]
```

```
[0.3919342961510144,
 0.1774186681885638,
 0.12318280005622713,
 0.11761584619506824,
 0.0916723817527774,
 0.07362229548013303,
 0.01683610289895436,
 0.007717609277261776]
```

```
pca_train = pca.fit_transform(X_train)

np.shape(pca_train)
```

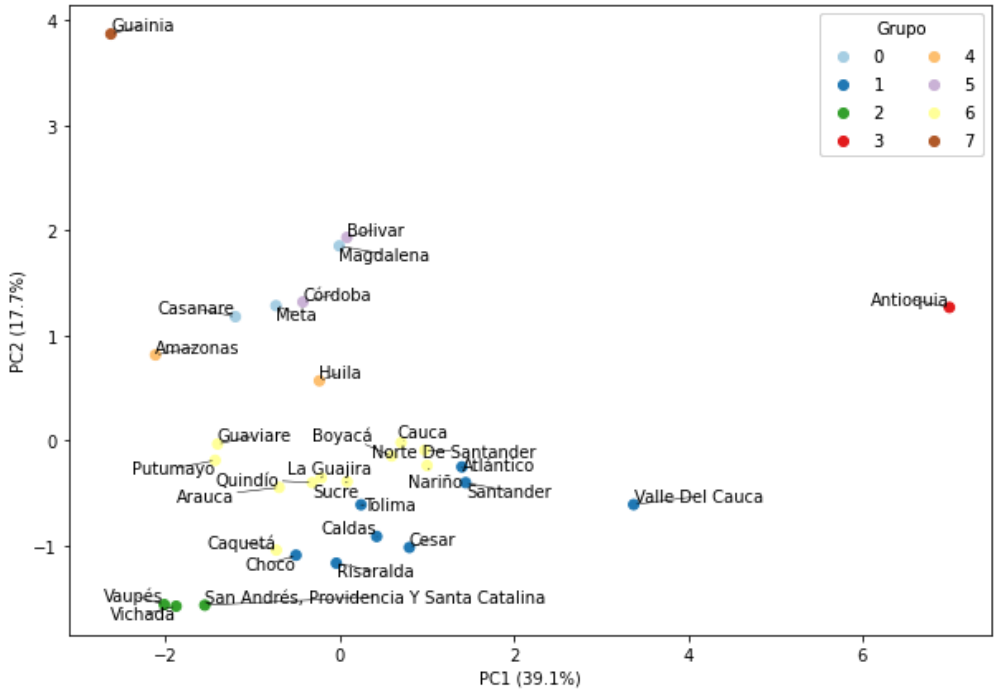
```
(31, 8)
```

```
pca_train1 = pd.DataFrame(pca_train, columns=None)
pca_train1.columns = ["PC{}".format(i+1) for i in range(8)]
pca_train1['Depto1'] = X.index
pca_train1['grupo'] = prediction
```

```
import pickle
```

```
fig, ax = plt.subplots(1,1, figsize = (10,7))

grupos = [int(i) for i in pca_train1.grupo]

scatter = ax.scatter(pca_train[:, 0], pca_train[:, 1], c = grupos,
           cmap='Paired')

texts = []
for i, label in enumerate(pca_train1['Depto1']):
  texts.append(ax.annotate(label, (pca_train[i,0], pca_train[i,1])))
```

```
adjust_text(texts, force_points=0.2, force_text=0.2,
            expand_points=(1, 1), expand_text=(1, 1),
            arrowprops=dict(arrowstyle="-", color='black', lw=0.5))

ax.set_xlabel('PC1 (39.1%)')
ax.set_ylabel('PC2 (17.7%)')
# ax.legend(grupos, [str(i) for i in grupos], loc = 'upper right', ncol = 2)

legend1 = ax.legend(*scatter.legend_elements(),
                    loc="upper right", title="Grupo", ncol = 2)
ax.add_artist(legend1)

plt.savefig('030_analisisCluster.pdf')
pickle.dump(fig, open('030_analisisCluster.pkl', 'wb'))
#
#
```



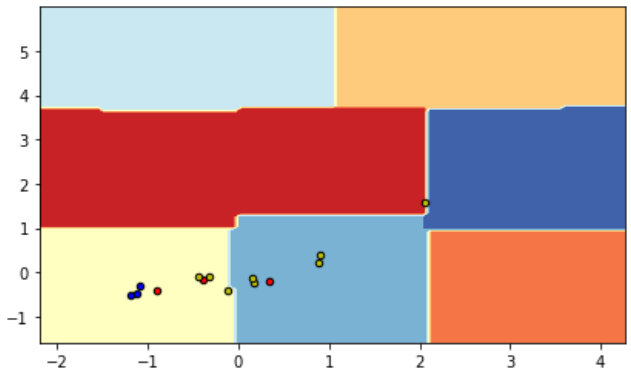
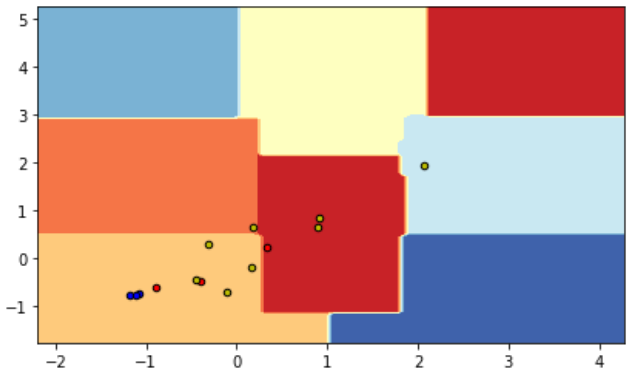

```
pca_train1.to_csv('030_analisisCluster.csv', index = False)
```

```
np.shape(X_train)

tree.fit_predict(X_train)
```

```
array([6., 2., 0., 3., 5., 3., 0., 0., 1., 3., 3., 0., 5., 4., 0., 4., 0.,
       1., 4., 3., 3., 7., 0., 0., 7., 3., 3., 0., 3., 7., 7.])
```

```
plot_n = 100
n_classes = 8

y = prediction

fig, _ = plt.subplots(1, 2, figsize = (15,4))

for pairidx, pair in enumerate([[0, 1], [0, 2]]):
    # Se toman sólo dos características
    Xi = scaler.transform(X)[:, pair]

    plt.subplot(1, 2, pairidx + 1)

    x_min, x_max = Xi[:, 0].min() - 1, Xi[:, 0].max() + 1
    y_min, y_max = Xi[:, 1].min() - 1, Xi[:, 1].max() + 1

    xx, yy = np.meshgrid(
        np.linspace(x_min, x_max, plot_n),
        np.linspace(y_min, y_max, plot_n)
    )

    Xj = np.zeros((plot_n**2, 8))

    rav_mat = np.c_[xx.ravel(), yy.ravel()]

    Xj[:, pair[0]] = rav_mat[:, 0]
    Xj[:, pair[1]] = rav_mat[:, 1]

    # print(np.shape(xx))
    # print(np.shape(Xj))
    # plt.tight_layout(h_pad=0.5, w_pad=0 .5, pad=2.5)

    Z = tree.fit_predict(Xj)
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        plt.scatter(
            Xi[idx, 0],
            Xi[idx, 1],
            c=color,
            # label=iris.target_names[i],
            cmap=plt.cm.RdYlBu,
            edgecolor="black",
            s=20,
        )

    # Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
    # plt.xlabel(iris.feature_names[pair[0]])
    # plt.ylabel(iris.feature_names[pair[1]])
```

```
Z
```

```
array([[4., 4., 4., ..., 1., 1., 1.],
       [4., 4., 4., ..., 1., 1., 1.],
       [4., 4., 4., ..., 1., 1., 1.],
       ...,
       [6., 6., 6., ..., 7., 7., 7.],
       [6., 6., 6., ..., 7., 7., 7.],
       [6., 6., 6., ..., 7., 7., 7.]])
```

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02

# Load data
iris = load_iris()

for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]):
    # We only take the two corresponding features
    X = iris.data[:, pair]
    y = iris.target

    # Train
```