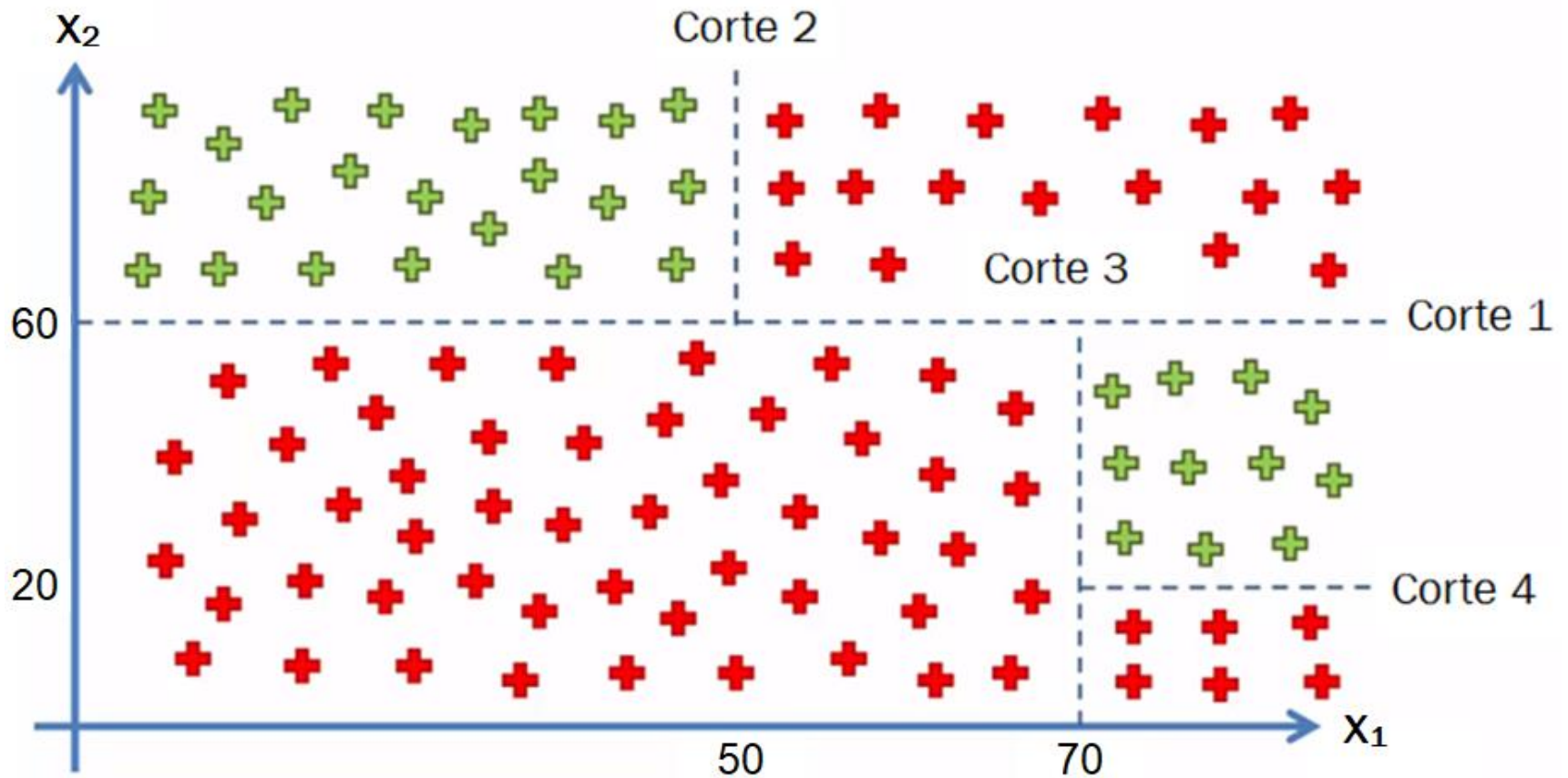
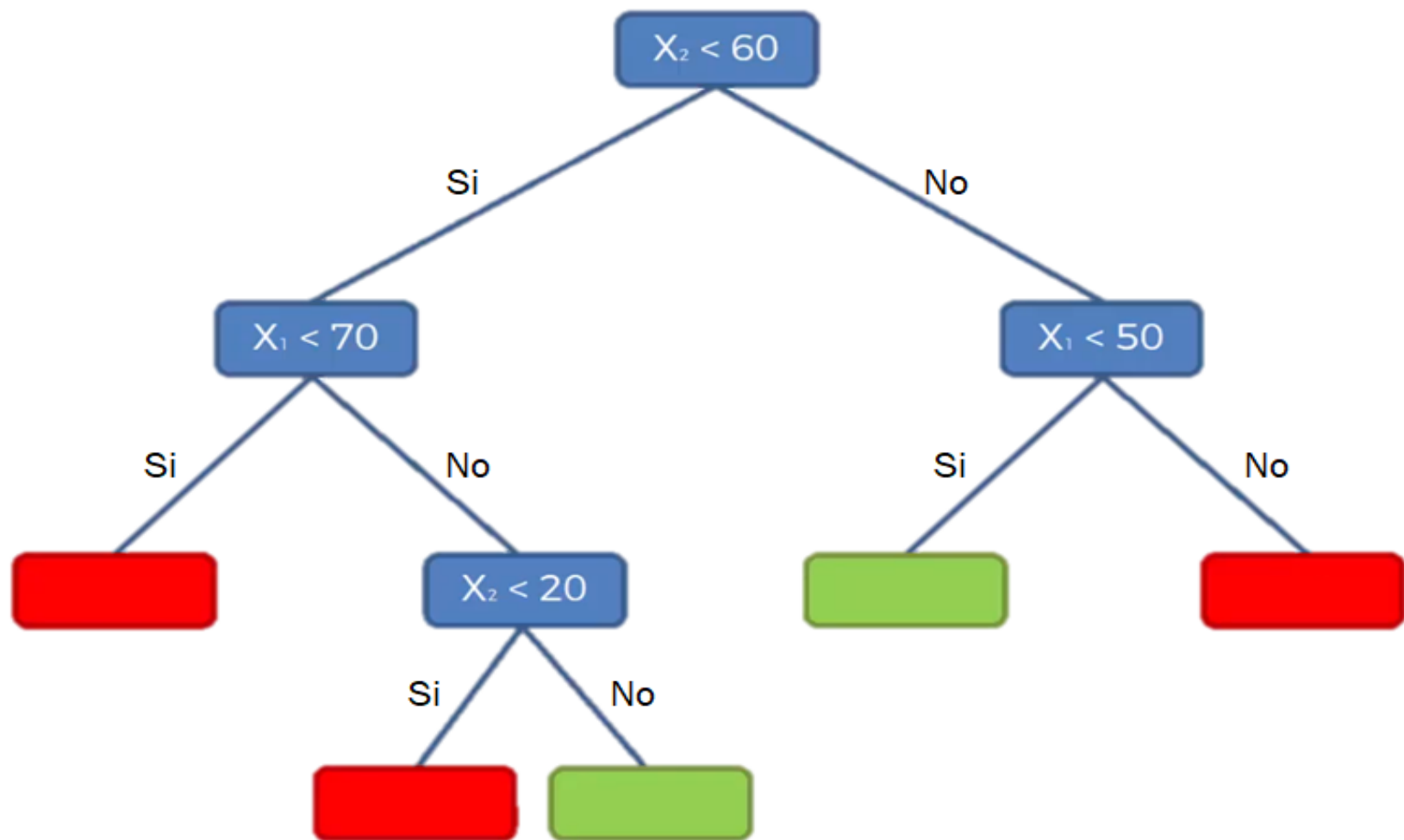





**CLUB DE DESARROLLO  
DE VIDEOJUEGOS  
E INTELIGENCIA ARTIFICIAL**

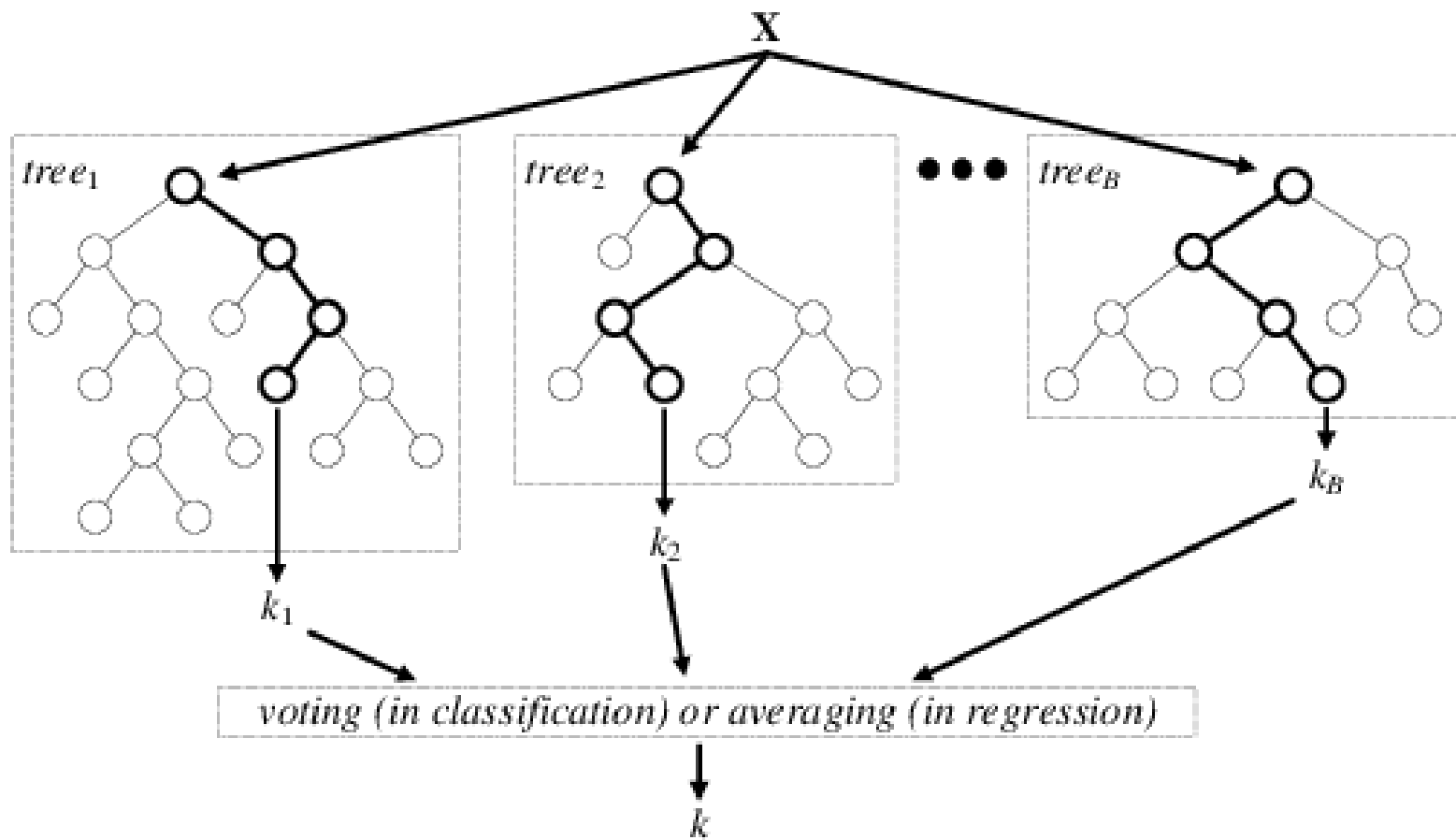
# ÁRBOLES DE DECISIÓN






# RANDOM FOREST

- Escoger de manera aleatoria  $k$  puntos del Training set.
  - Crear un árbol de decisión asociado a esos puntos.
  - Repetir  $n$  veces los pasos anteriores donde  $n$  es el número de árboles.
  - Para el nuevo punto ejecutar los  $n$  árboles y ver la categoría más votada.
- 




# RANDOM FOREST EN PYTHON

- Pre-procesamiento de datos.
  - Ajustar nuestro clasificador a nuestro train set.
  - Predecir los valores del test set.
  - Visualizar los resultados.
- 

# PRE-PROCESAMIENTO DE DATOS

```
dataset = pd.read_csv('Anuncios_redes.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
    0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```



# AJUSTE DE REGRESOR

- Importar la clase para random forest.


```
from sklearn.ensemble import RandomForestClassifier
```

- Crear un objeto de la clase con los parámetros necesarios.

```
clasificador = RandomForestClassifier(n_estimators = 10, criterion =  
    'entropy', random_state = 0)
```

- Aplicar los métodos de la clase a nuestros datos.

```
clasificador.fit(X_train, y_train)
```



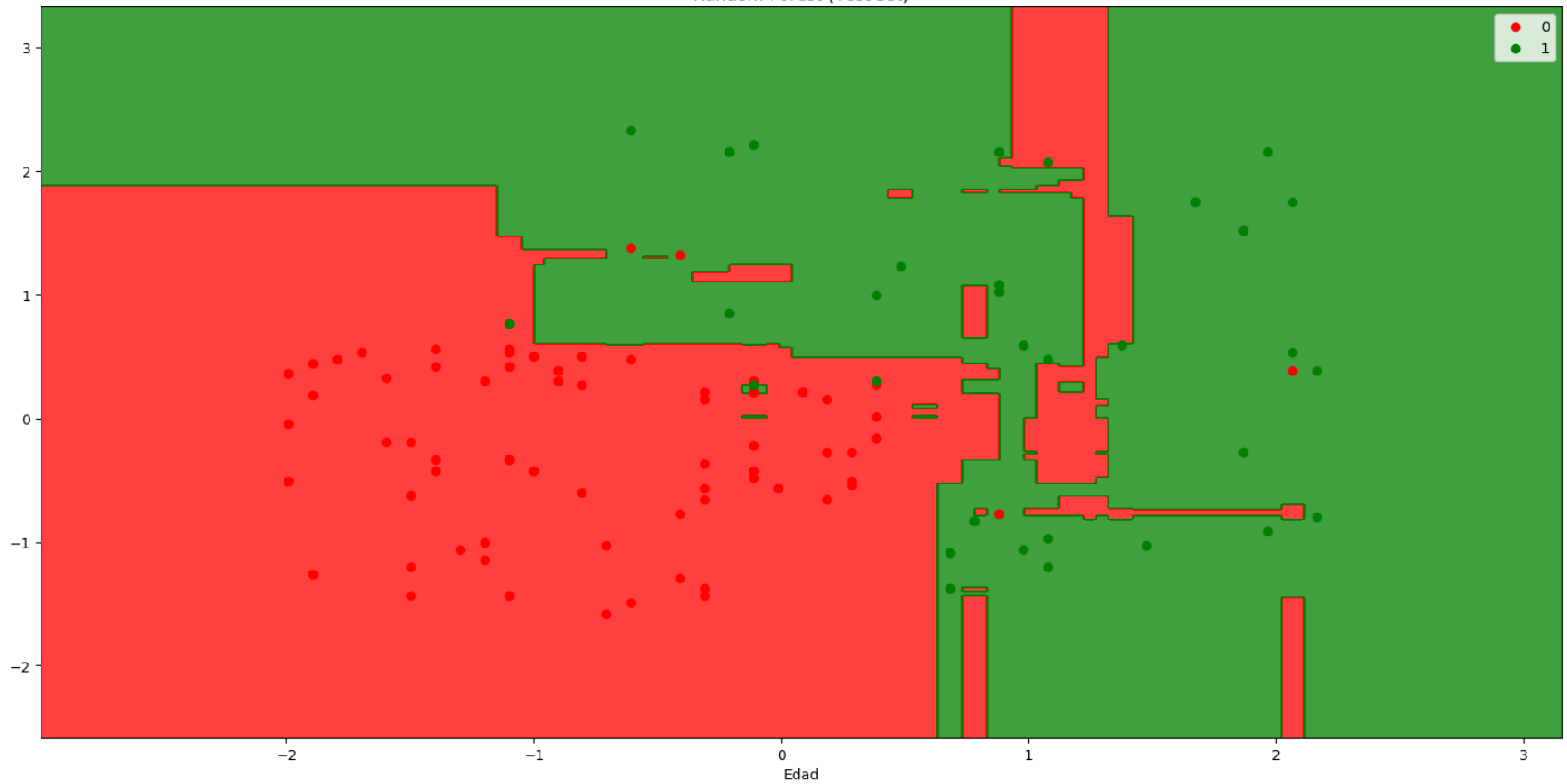


# REALIZACIÓN DE PREDICCIONES


```
y_pred = clasificador.predict(X_test)
```

# VISUALIZACIÓN DE RESULTADOS

Random Forest (Test set)



# OPTIMIZACIÓN DE HIPERPARÁMETROS

- Importar la clase para optimización de hiperparámetros.
  - Crear una lista de diccionarios con todos los valores de los hiperparámetros que se quieren testear.
  - Crear un objeto de la clase con los parámetros necesarios.
  - Ajustar el optimizador a nuestros datos.
  - Extraer la mayor precisión y los mejores parámetros.
- 

# OPTIMIZACIÓN DE HIPERPARÁMETROS EN PYTHON

```
from sklearn.model_selection import GridSearchCV

parametros_knn = [{'n_neighbors': [5, 10, 20, 50], 'algorithm':
    ['ball_tree', 'kd_tree', 'auto'], 'leaf_size': [30, 50, 100], 'p': [1, 2],
    'metric': ['minkowski']}]

grid_search_knn = GridSearchCV(estimator = clasificador_knn,
    param_grid = parametros_knn, scoring = 'accuracy', cv = 10,
    n_jobs = -1)

grid_search_knn = grid_search_knn.fit(X_train, y_train)

best_accuracy_knn = grid_search_knn.best_score_

best_parameters_knn = grid_search_knn.best_params_
```