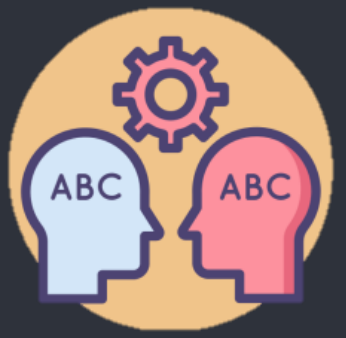
# Procesamiento de Lenguaje Natural{




[NLP]

<Aprendizaje Supervisado | Clasificacion | Decision Trees>

}

Ivan Rojas Gonzalez – Universidad Panamericana

# La Agenda de hoy {

## 01 Intuicion sobre Decision Trees

<Cómo funciona>

## 02 Implementando Decision Trees

<Llevando nuestro clasificador a la vida>

## 03 Evaluando el clasificador

<Que tan bueno es nuestro clasificador?>

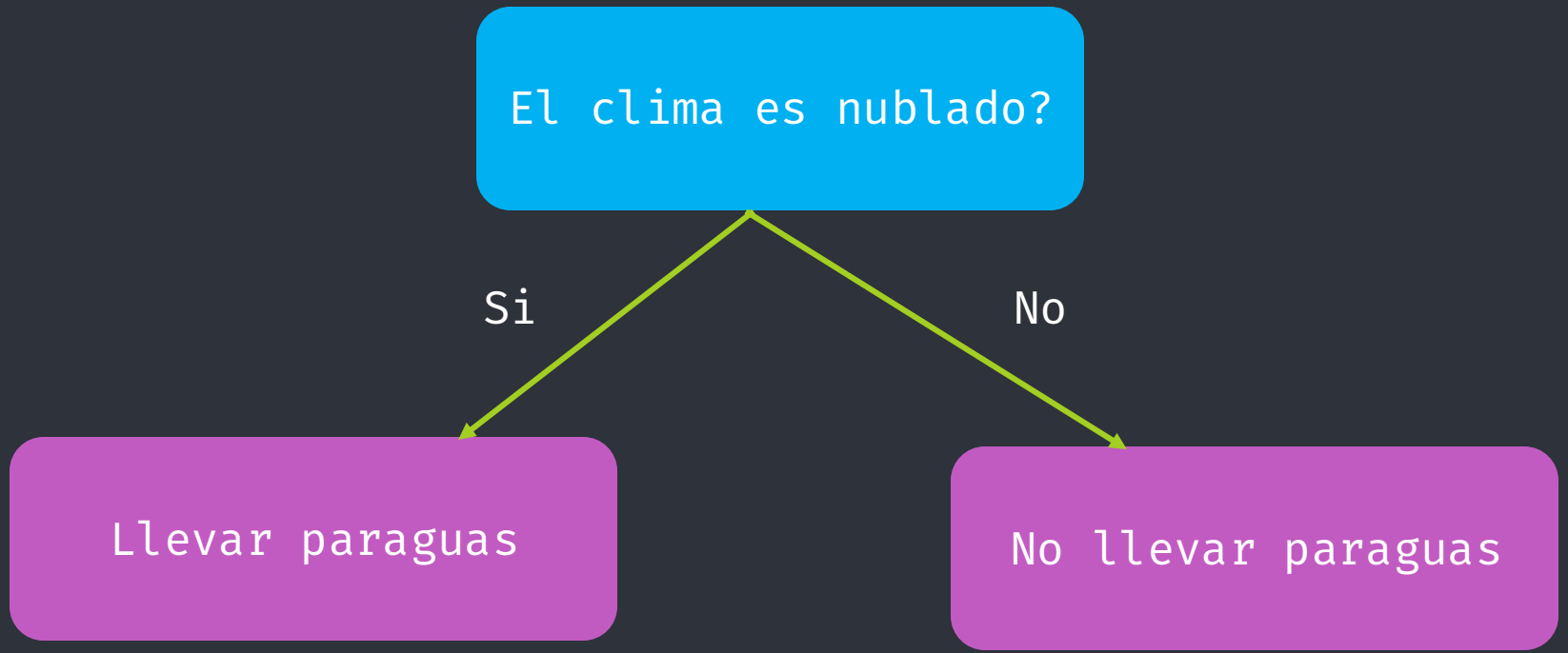
}

# {Decision Trees}

Ivan Rojas Gonzalez – Universidad Panamericana

# Decision Trees{

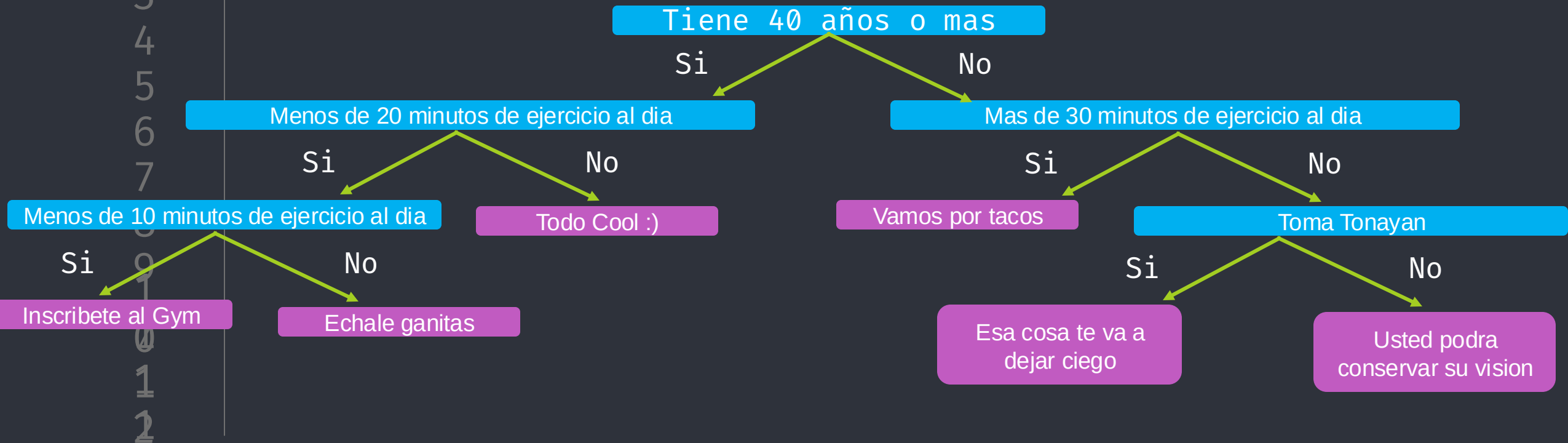
<Nos centraremos en arboles de decisión para clasificación>




}

# Decision Trees{

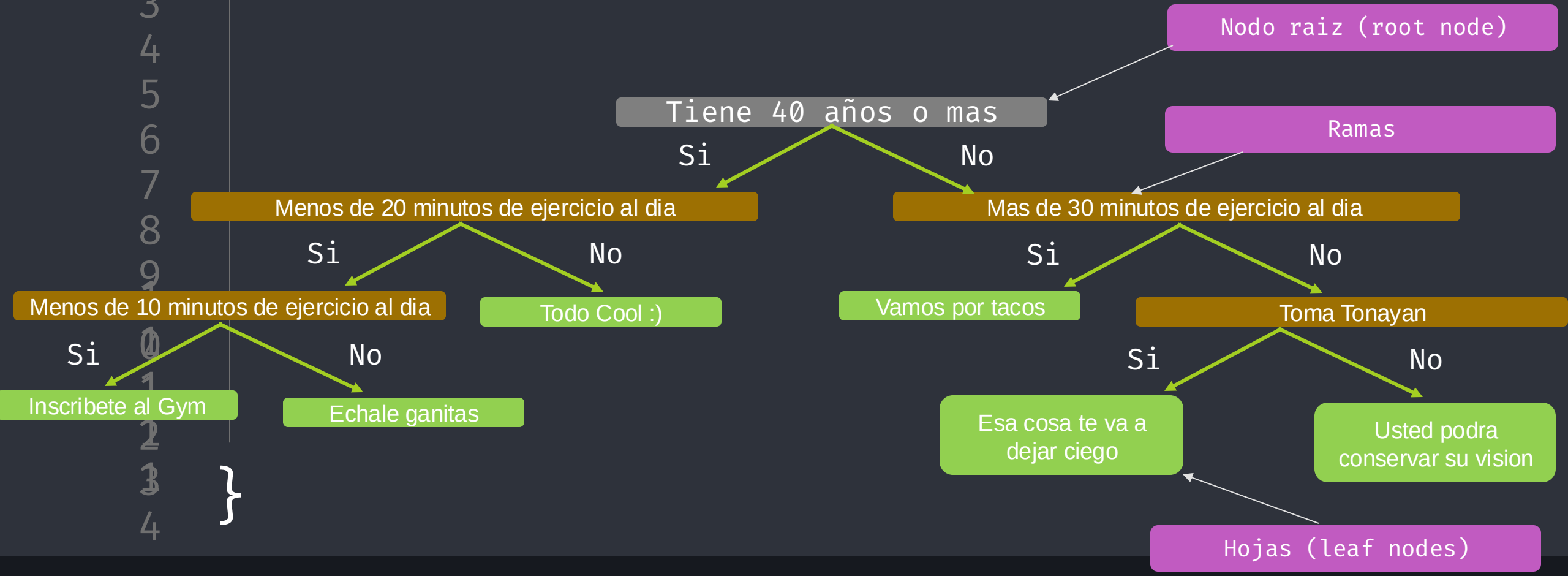
<Nos centraremos en arboles de decisión para clasificación>




}

# Decision Trees{

<Nos centraremos en arboles de decisión para clasificación>

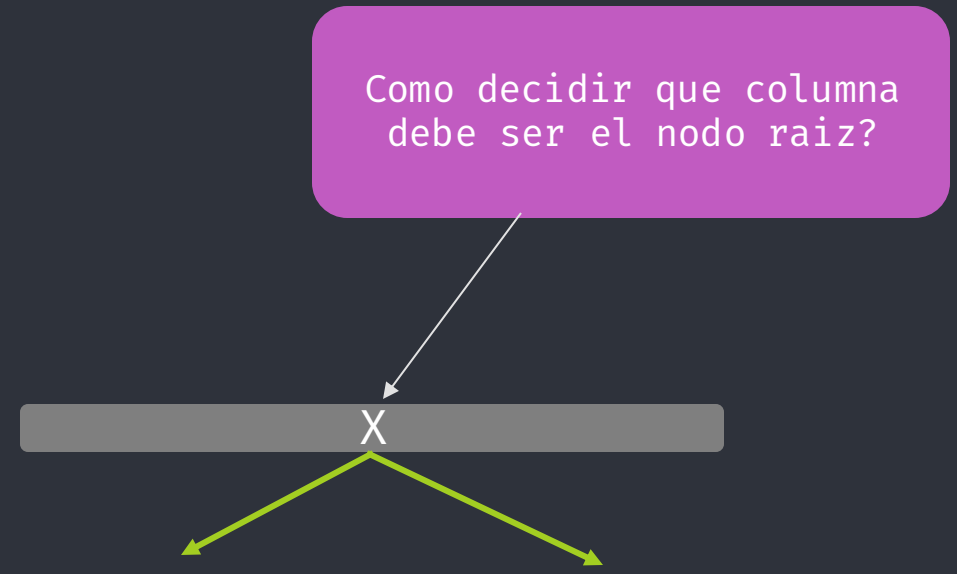



}

# Decision Trees{

| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

}

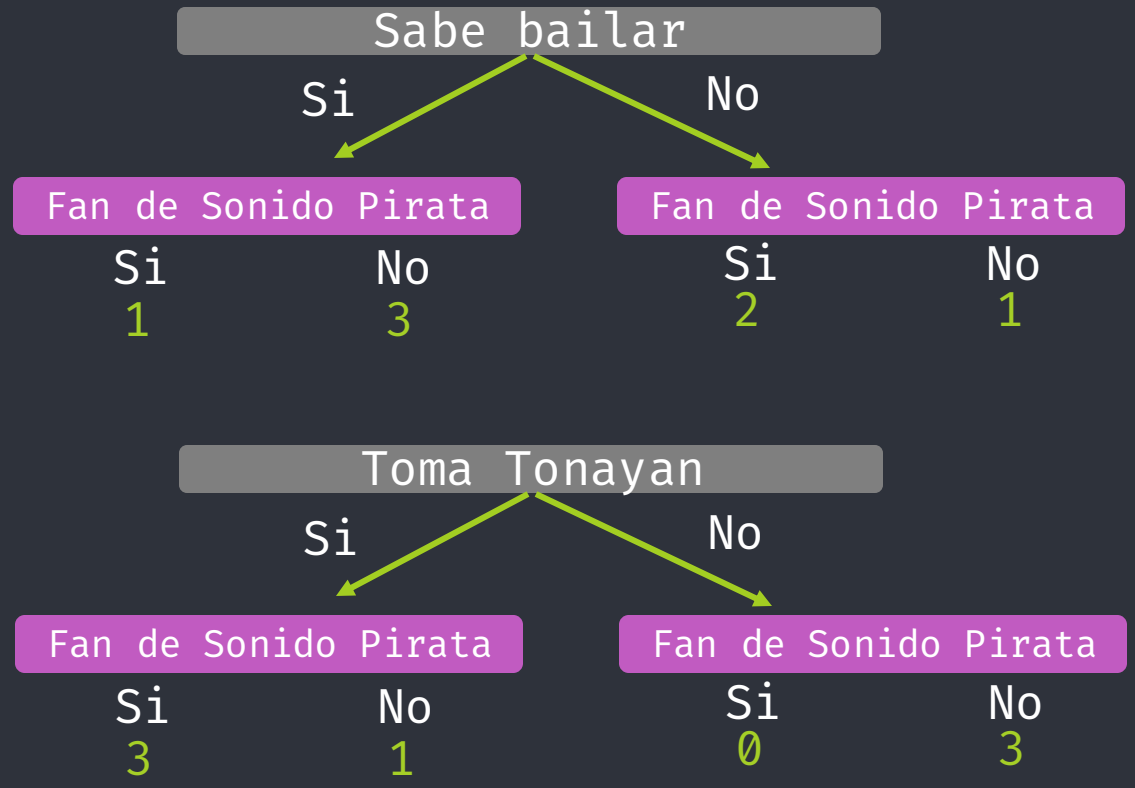


# Decision Trees{

| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

}

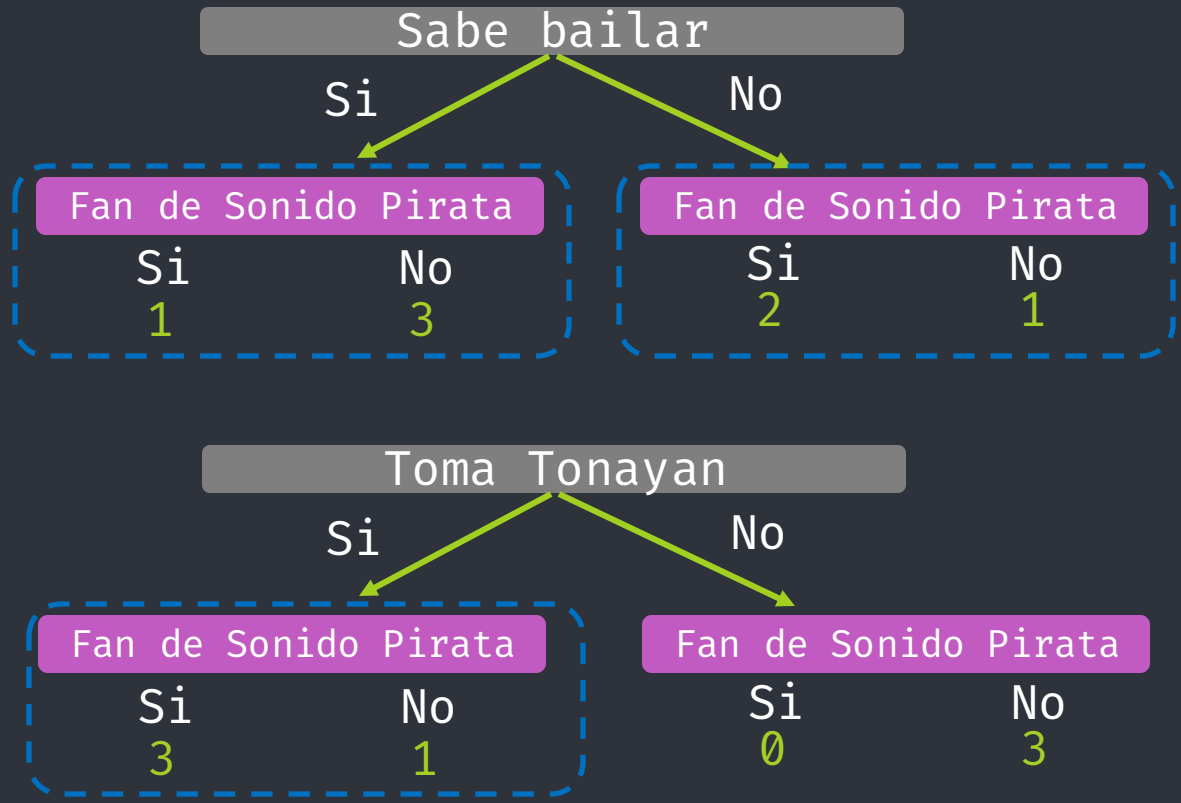
<Evaluar columna por columna para saber cual predice mejor>



# Decision Trees{

Leaf Nodes Impuros

<Evaluar columna por columna para saber cual predice mejor>




}

# Decision Trees{

Gini impurity

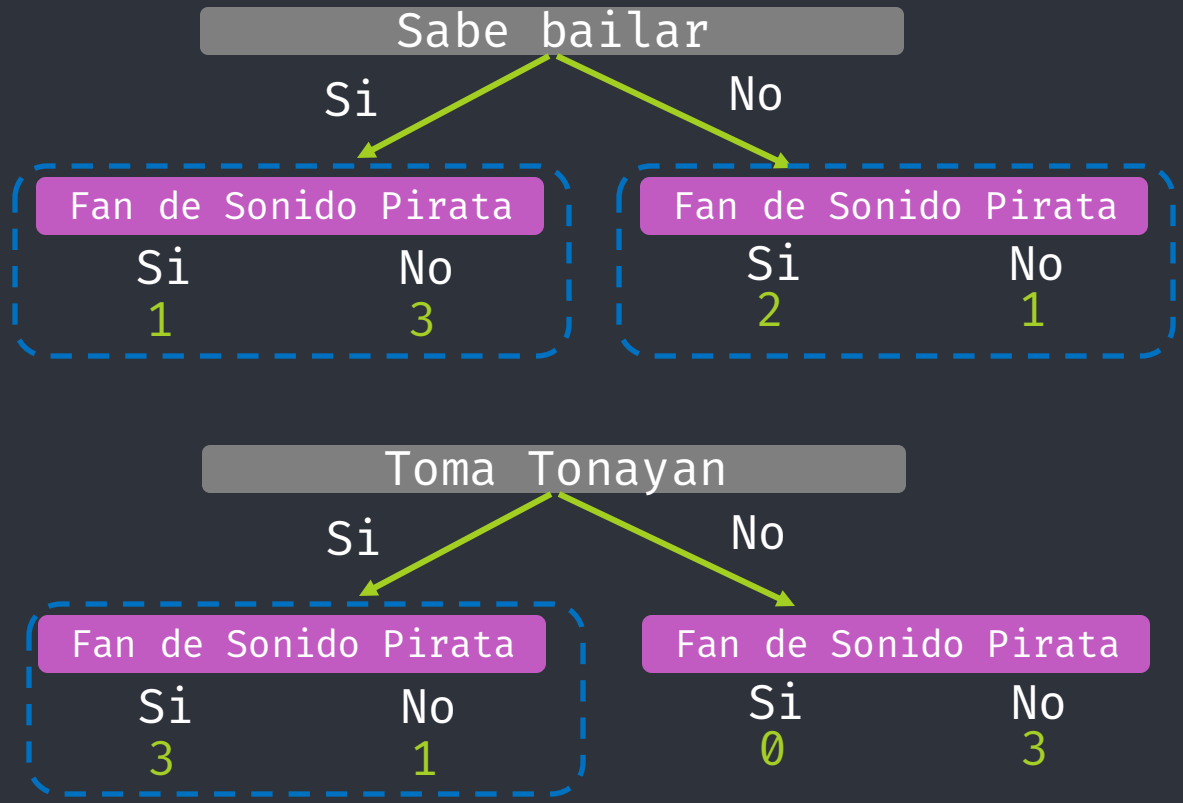
$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

$$GI = 1 - \left[ (P_{(+)})^2 + (P_{(-)})^2 \right]$$

Se calcula por cada leaf node

}

<Evaluar columna por columna para saber cual predice mejor>



# Decision Trees{

## Gini impurity

$$GI = 1 - \left[ (P_{(+)})^2 + (P_{(-)})^2 \right]$$

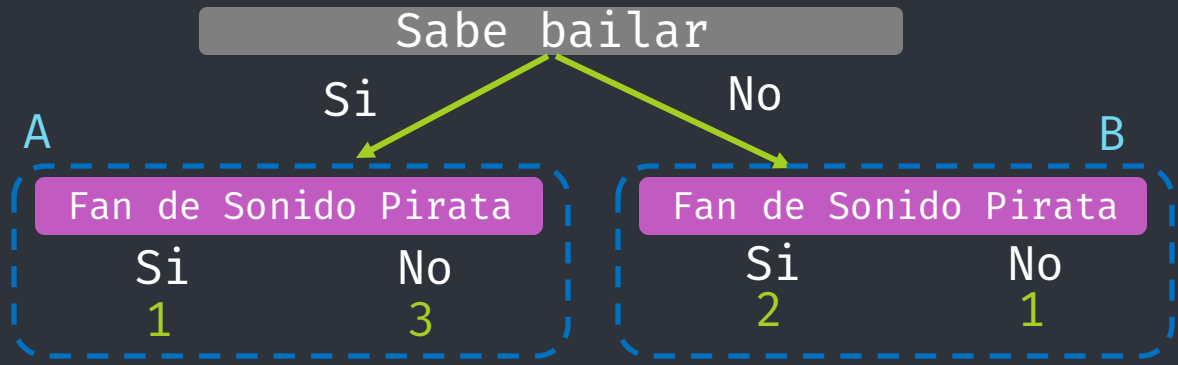
Gini Impurity para leaf node A

1 - (1/(1+3))^2 - (3/(1+3))^2

1 - (1/4)^2 - (3/4)^2

0.375

}



# Decision Trees{

## Gini impurity

$$GI = 1 - \left[ (P_{(+)})^2 + (P_{(-)})^2 \right]$$

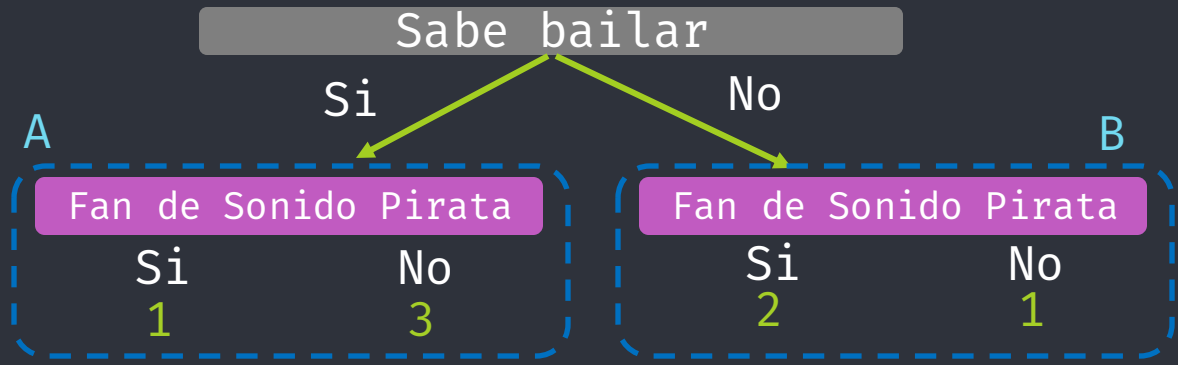
Gini Impurity para leaf node B

1 - (2/(2+1))^2 - (1/(2+1))^2

1 - (2/3)^2 - (1/3)^2

0.444

}



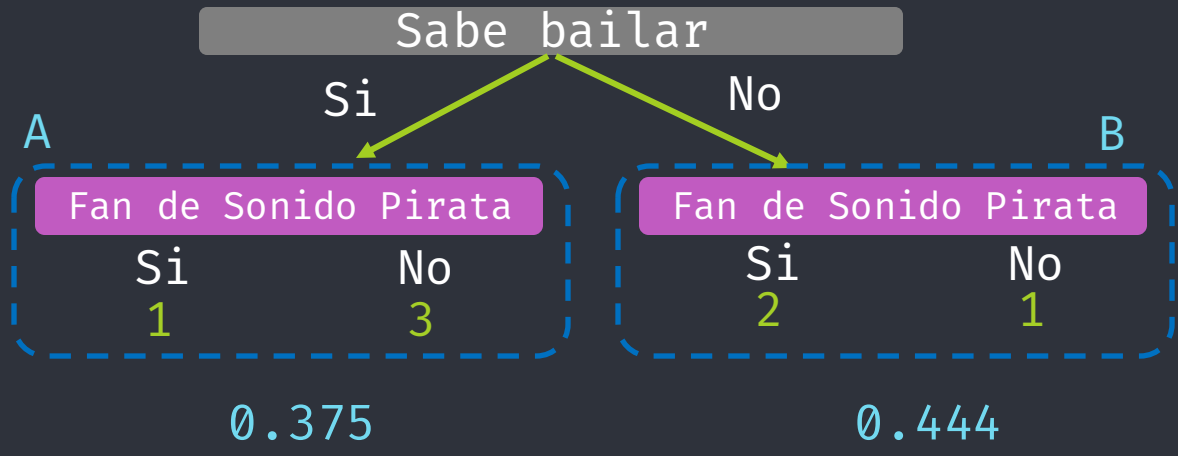
# Decision Trees{

## Gini impurity

Gini Impurity Total
sera la media ponderada

0.375 * (4/7) + 0.444*(3/7)

**0.405**

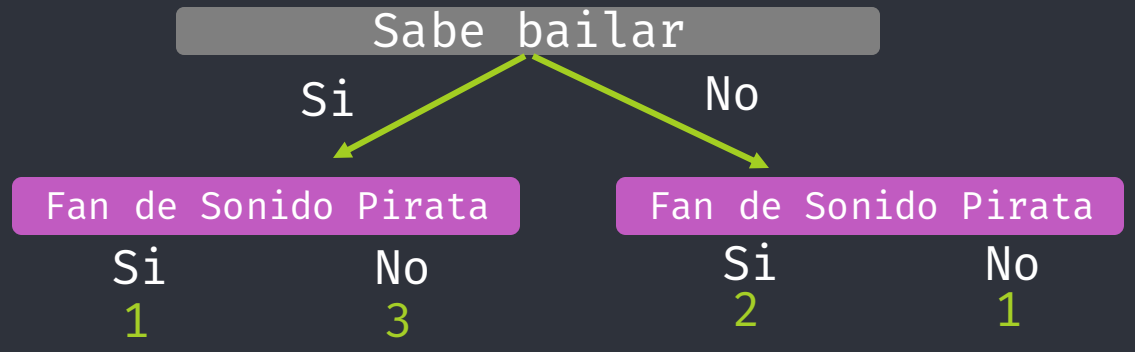



}

# Decision Trees{

Gini impurity

**0.405**

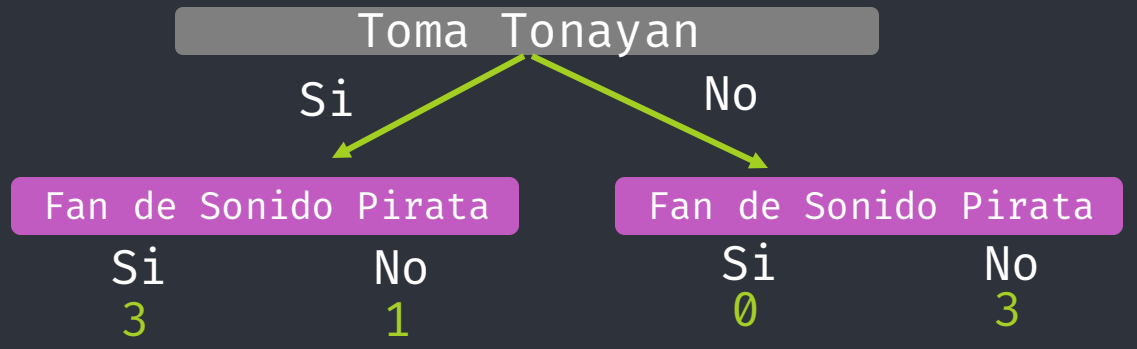



}

# Decision Trees{

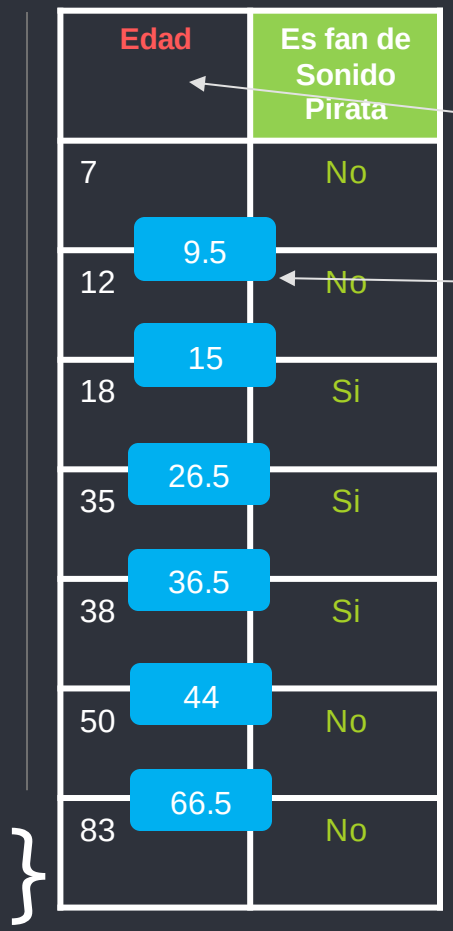
Gini impurity

**0.214**




}

# Decision Trees{




<Gini impurity de datos numericos>

Se acomodan los valores de menor a mayor

Se calcula el promedio para cada par

Se calcula gini impurity por cada valor

Edad < 9.5

Si

No

Fan de Sonido Pirata

Si 0

No 1

Giny impurity = 0

Fan de Sonido Pirata

Si 3

No 3

Giny impurity = 0.5

Giny impurity total = 0.429

# Decision Trees{

<Gini impurity de datos numericos>

| Edad | Es fan de Sonido Pirata |
|---|---|
| 7 | No |
| 12 | No |
| 18 | Si |
| 35 | Si |
| 38 | Si |
| 50 | No |
| 83 | No |

9.5 | 15 | 26.5 | 36.5 | 44 | 66.5

}

Giny impurity total = 0.429

Giny impurity total = 0.343

Giny impurity total = 0.476

Giny impurity total = 0.476

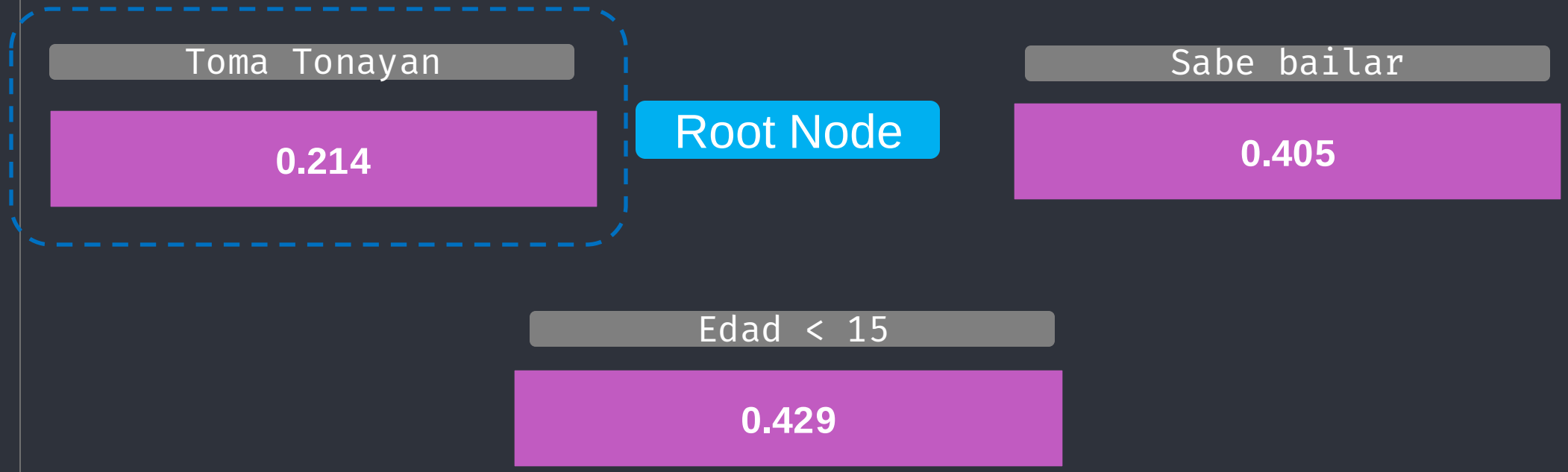
Giny impurity total = 0.343

Giny impurity total = 0.429

Escogemos el más bajo. En caso de empate, se elegimos cualquiera de los empatados

# Decision Trees{

<Comparamos Giny Impurity de las 3 columnas>

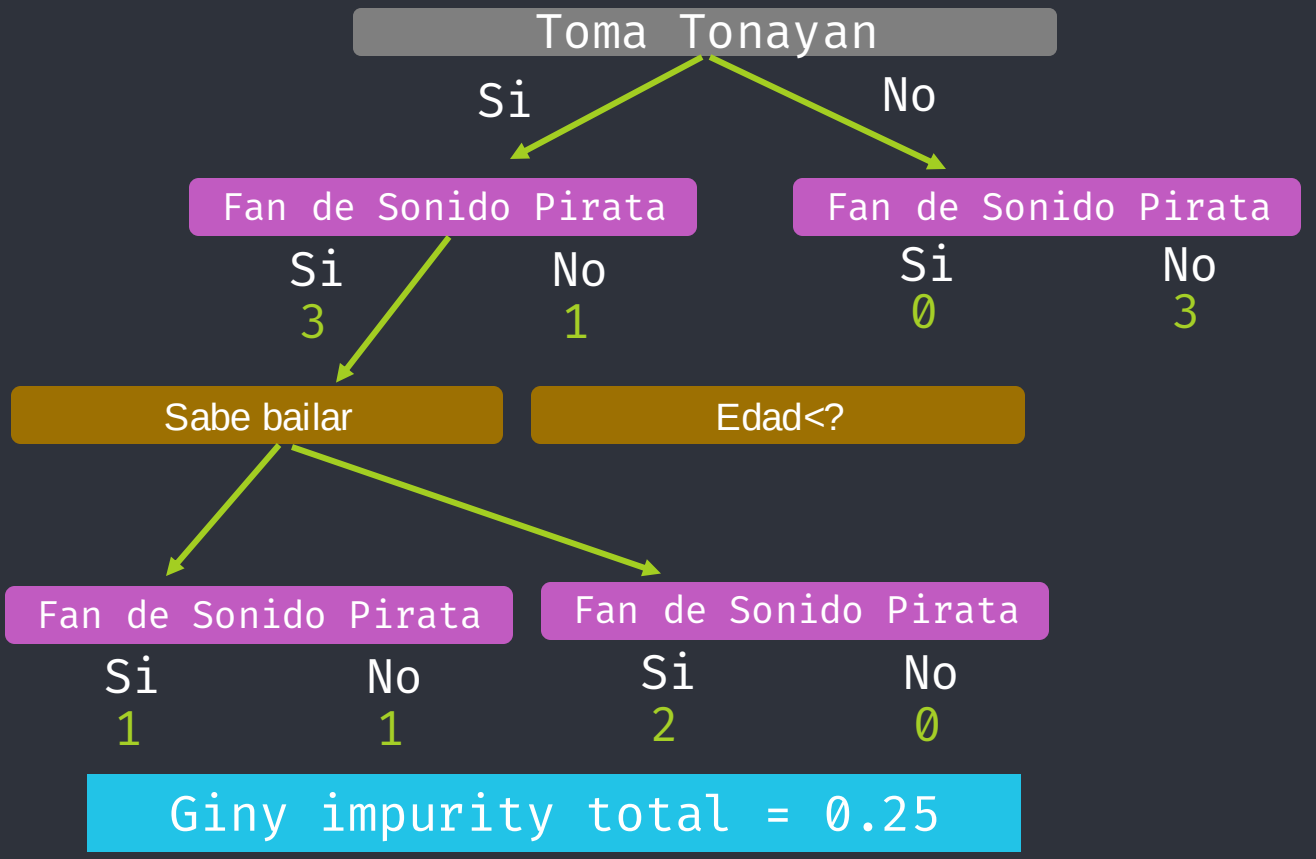



}

# Decision Trees{

| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

}



# Decision Trees{

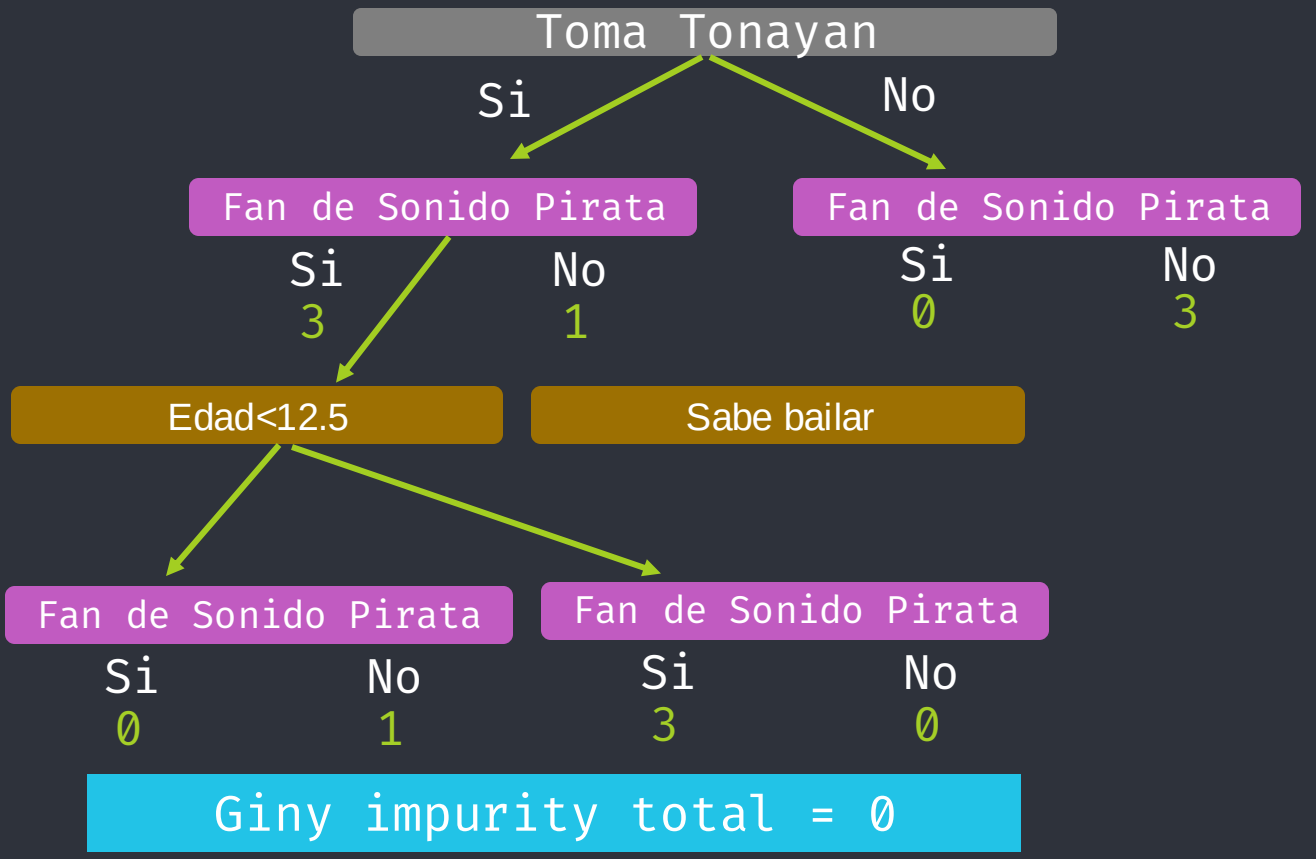
| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

12.5

26.5

36.5

}



# Decision Trees{

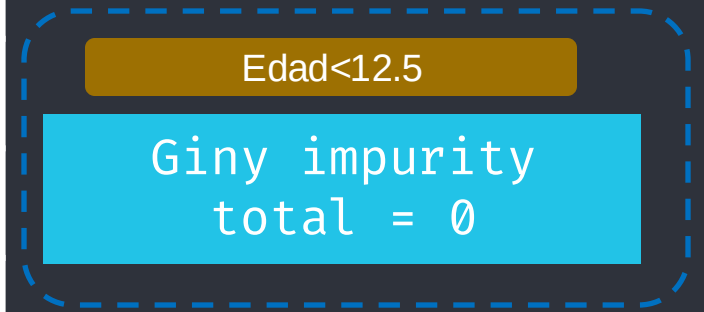
| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

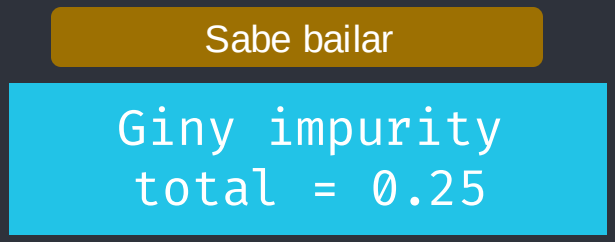
12.5

26.5

36.5

}

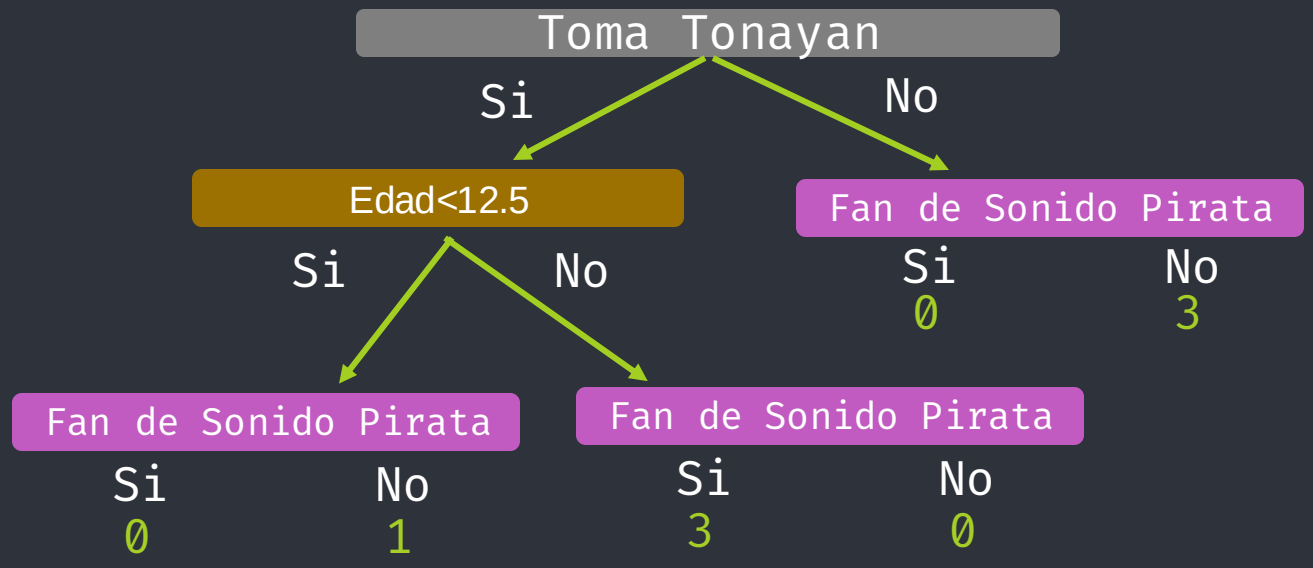


# Decision Trees{

| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

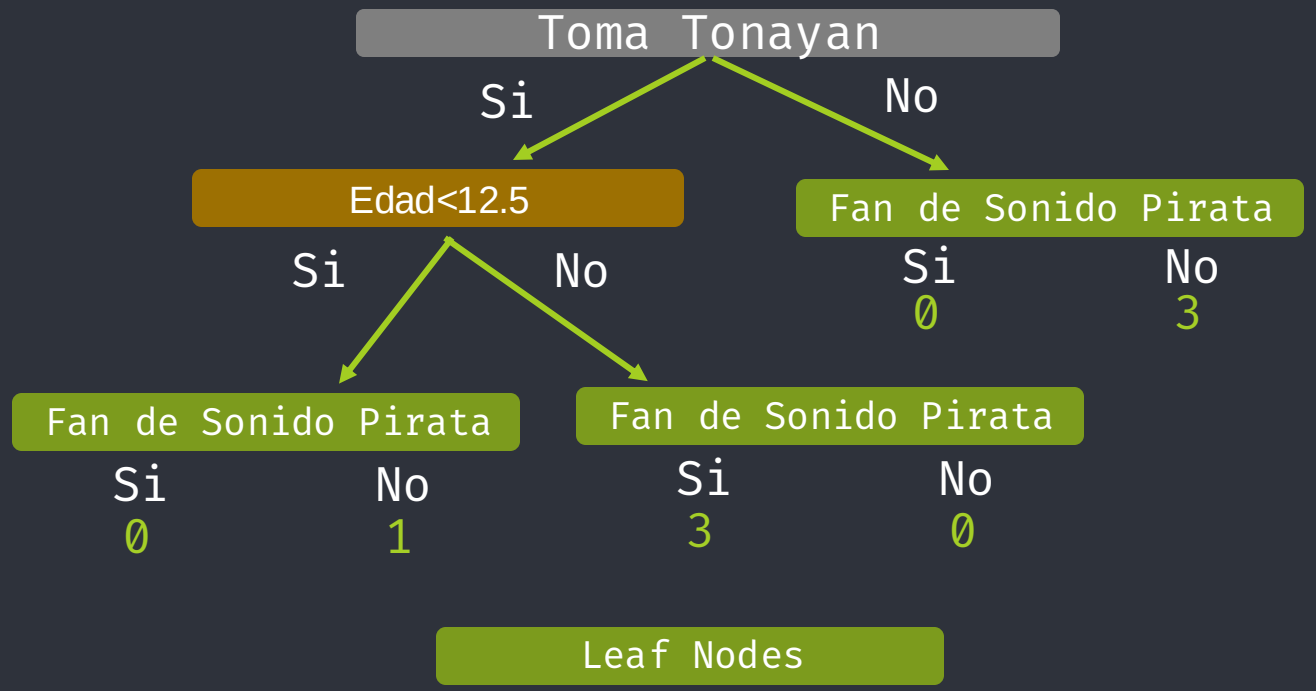
}



# Decision Trees{

| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

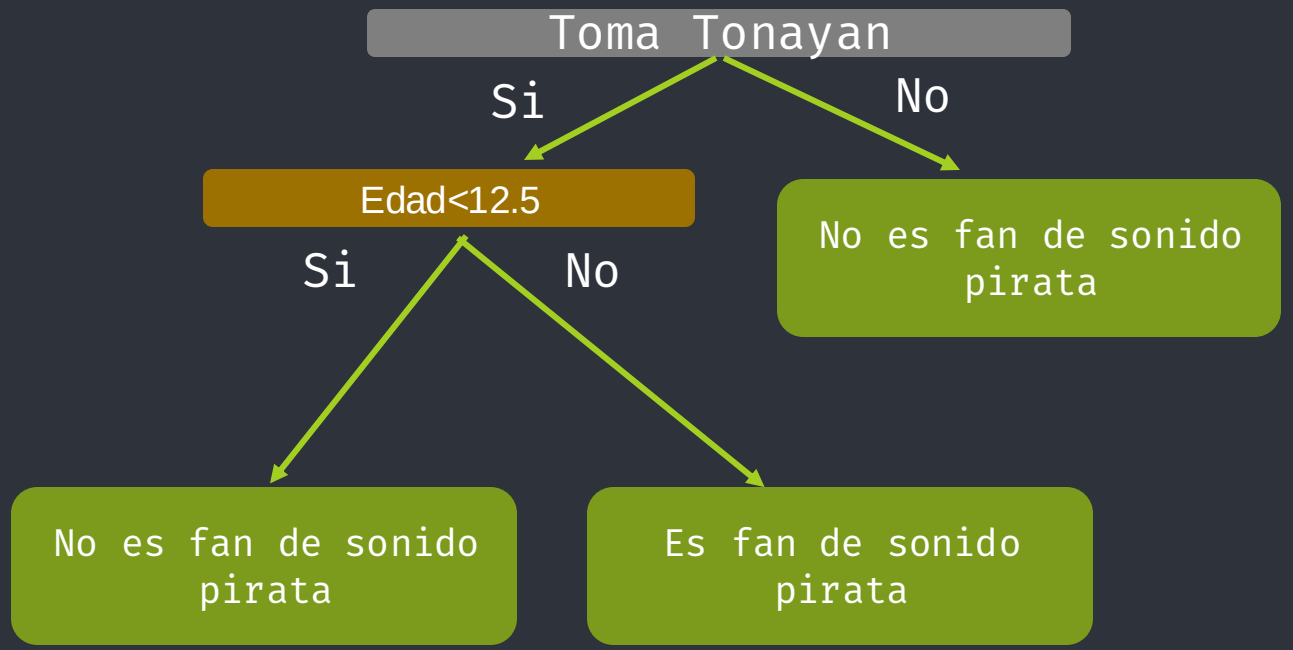
}



# Decision Trees{

| Sabe Bailar | Toma Tonayan | Edad | Es fan de Sonido Pirata |
|---|---|---|---|
| Si | Si | 7 | No |
| Si | No | 12 | No |
| No | Si | 18 | Si |
| No | Si | 35 | Si |
| Si | Si | 38 | Si |
| Si | No | 50 | No |
| No | No | 83 | No |

}



# {Implementando Decision Trees}

Ivan Rojas Gonzalez – Universidad Panamericana

# Implementando Decision Trees{

<Dado que estamos tratando con aprendizaje supervisado, tendremos que partir los datos en un set de entrenamiento y un set de prueba>

```python
from sklearn.model_selection import train_test_split

X = df['tweet_clean']
y = df['intention']

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.85)
```

Se importa la libreria

Variable(s) de entrenamiento

Variable objetivo

% de los datos reservados para entrenamiento

}

# Implementando Decision Trees{

<Ahora procedemos a vectorizar el texto>

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(ngram_range=(1,3))

X_train_vectorized = tfidf.fit_transform(X_train)
X_test_vectorized = tfidf.transform(X_test)
```

Usamos fit_transform para los datos de entrenamiento

Y transform para los datos de prueba

}

# Implementando Decision Trees{

<Ahora, creamos nuestro modelo de decision trees>

```python
from sklearn.tree import DecisionTreeClassifier
model= DecisionTreeClassifier()
model.fit(X_train_vectorized, y_train)
y_pred = model.predict(X_test_vectorized)
```

Importamos la libreria de maquinas de soporte vectorial para clasificacion

Instanciamos un modelo

Y lo entrenamos sobre los datos de entrenamiento

Obtenemos las predicciones sobre el set de prueba para sacar metricas de clasificacion

}

# Implementando Decision Trees{

<Finalmente, evaluamos el desempeño>

```
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test,y_pred))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| clickbait | 0.84 | 0.93 | 0.88 | 6419 |
| non-clickbait | 0.93 | 0.82 | 0.87 | 6381 |
| accuracy | | | 0.88 | 12800 |
| macro avg | 0.88 | 0.88 | 0.88 | 12800 |
| weighted avg | 0.88 | 0.88 | 0.88 | 12800 |

Importamos componentes de la libreria de metricas

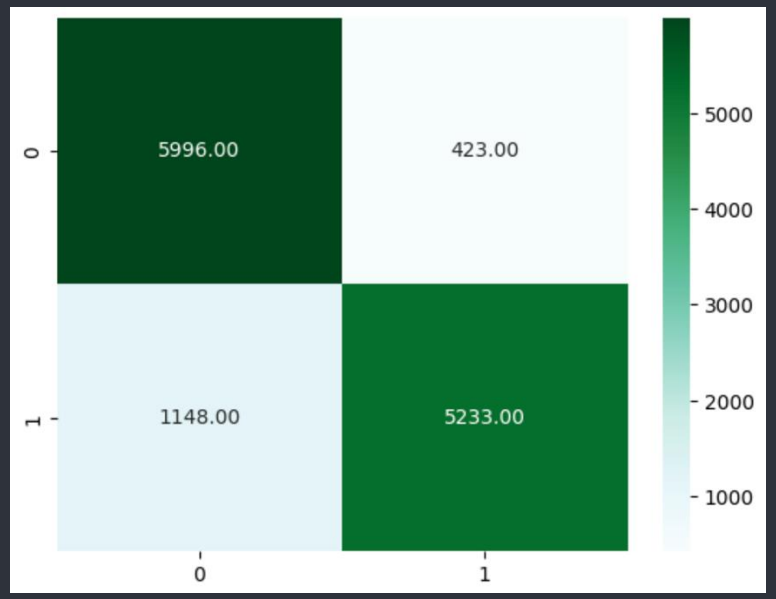
Obtenemos el reporte de clasificacion

}

# Implementando Decision Trees{

<De igual manera, podemos obtener la matriz de confusion>

```
import seaborn as sns
sns.heatmap(confusion_matrix(y_test,y_pred), annot=True, fmt='.2f')
```




}

1
2
3
4
5
6
7
8
9
10
11
12
13
14

# Q&A