{desafío}
latam_

Árboles de decisión _

Sesión Presencial 1

Itinerario

Activación de conceptos

Desarrollo Desafío

Panel de discusión

{desafío}
latam_

Activación de conceptos

¿Qué permite resolver EM?

- El problema de información faltante.
- La agrupación de datos en función de una distribución subyacente.
- El problema de variables latentes/datos perdidos.

¿Qué significa cada paso de EM?

- En **E** obtenemos la media y en **M** minimizamos respecto a la función de pérdida.
- En **E** evaluamos la media empírica y en **M** imputamos el mínimo estimable de la función candidata.
- En **E** obtenemos la esperanza y en **M** maximizamos la verosimilitud del modelo para encontrar un nuevo punto de actualización.

¿Qué informan los Criterios de Información de Akaike y Bayesiano?

- Minimización del riesgo empírico.
- La bondad de ajuste respecto a la función de pérdida.
- La bondad de ajuste respecto a la función de verosimilitud del modelo.

¿Cuál de los siguientes problemas no se puede resolver mediante EM?

- Variables latentes.
- Minimización de pérdida empírica.
- Imputación de los datos perdidos.

¿Cómo podemos solucionar el problema de la clasificación y regresión de forma eficiente, efectiva y a la vez utilizando un modelo que sea interpretable y fácil de explicar?

- Algoritmos como SVM, EM, GAM → Fáciles para la máquina, difíciles para nosotros.
- Objetivo → Aprender sobre un problema a partir de una jerarquía de preguntas en formato if/else.
- Esto es análogo a nuestro procesamiento mental de discriminar.

Intuición: Clasificando animales

```
In [2]: df.sample(2, random_state=11238)
```

Out[2]:

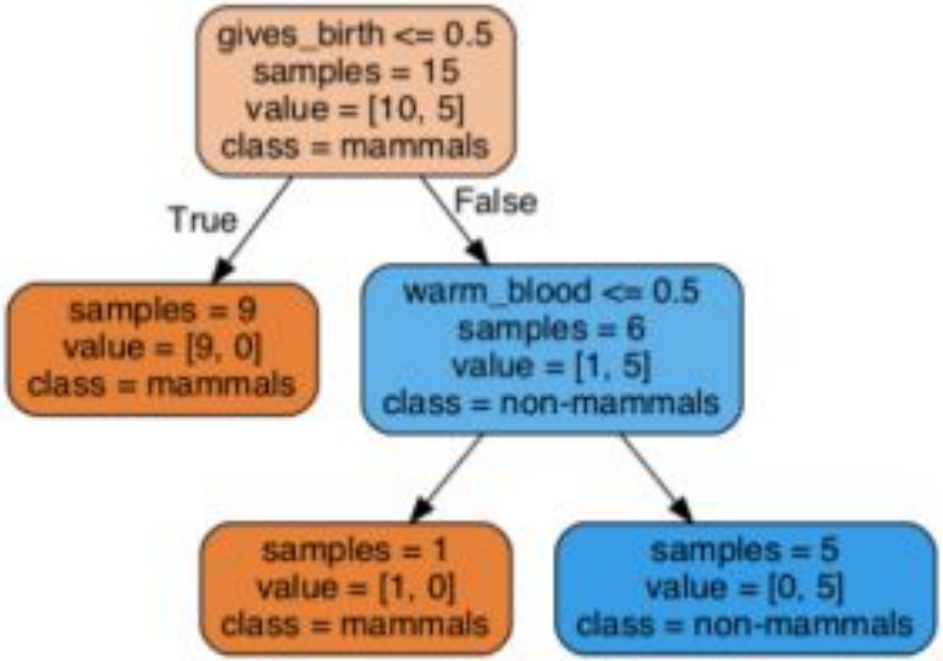
	name	warm_blood	gives_birth	aquatic	aerial	has_legs	hibernates	
8	cat	1	1	0	0	1	0	mam
2	salmon	0	0	1	0	0	0	fishe

- Objetivo → Aprender sobre las características de $y \in \{\text{Mamifero}, \neg\text{Mamifero}\}$
- ¿Cómo? → Identificar qué atributos $\mathbf{x}_p \in \mathbf{X}$ facilitan la discriminación en y.
- Esto se logra mediante la separación de clases dentro de cada pregunta → Optimización Local.

Taxonomía de un árbol

```
In [4]: Image(dec_tree.create_png())
```

Out[4]:



- Cuadros:
  - **Nodos**: Existen nodos principales y terminales
- Flechas:
  - **Ramas:** Establecen el flujo dado el problema de optimización local.

Problemas

- ¿Cómo diferenciamos entre atributos importantes?
- ¿Cómo seleccionamos el puntaje de partición?
- ¿Hasta qué punto dejamos crecer los árboles?

Aspectos

Optimizar un puntaje de corte en un nodo de forma local.

$$\hat{f}(x) = \sum_{m=1}^{s} c_m I\left((X_1, X_2) \in \mathbb{R}_m\right)$$

Donde:

- $\mathbb{R}_m \ni (\mathbf{X})$ es la región producto de la partición en el espacio de atributos.
- c_m es el promedio del vector objetivo en la región definida.
- **s** es el puntaje de corte que permite asignar una observación a una de las regiones.

Asignación

$$R_1(j,s) = \{X | X_j \leq s\} \quad \text{ó} \quad R_2(j,s) = \{X | X_j \geq s\}$$

Dado $R \geq 2$, asignar la nueva observación a la región en base a su puntaje calculado.

Optimización del puntaje de corte

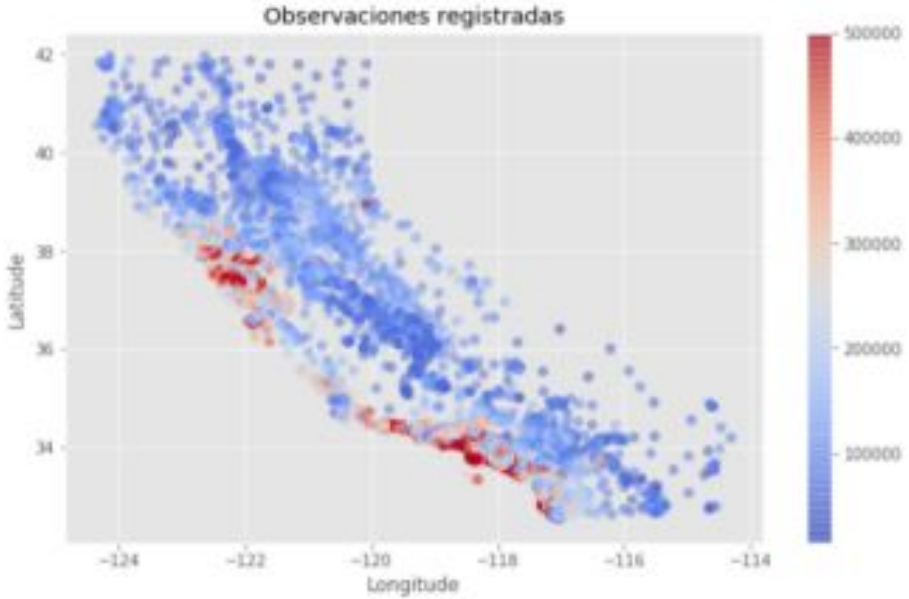
- El problema sigue siendo que no sabemos cómo identificar **s**.
- Resolvemos el siguiente problema de optimización.

$$\underset{j,s}{\text{argmin}}\left[\underset{c_1}{\text{argmin}}\sum_{x_i\in R}(y_i-c_1)^2+\underset{c_2}{\text{argmin}}\sum_{x_i\in R}(y_i-c_2)^2\right]$$

Árboles de regresión

Objetivo: Predecir precios en California

```
In [6]: plt.scatter(df['Longitude'], df['Latitude'],c = df['MedianHouseValue'],cmap='coolwarm',al
pha=.5)
plt.colorbar(); plt.xlabel('Longitude'); plt.ylabel('Latitude'); plt.title('Observaciones
 registradas');
```



{desafío} latam_

sklearn.tree.DecisionTreeRegressor

Pasos clásicos:

- **Preprocesar**: Utilizamos **`np.log`** para dos atributos.
- **Segmentar**: Utilizamos **`sklearn.model_selection.train_test_split`**
- Entrenar

```
In [8]: from sklearn.tree import DecisionTreeRegressor
        dec_tree = DecisionTreeRegressor().fit(X_train, y_train)

In [9]: from sklearn.metrics import mean_squared_error, median_absolute_error, r2_score
        print("Test MSE:", mean_squared_error(y_test, dec_tree.predict(X_test)).round(5))
        print("Test MAE:", median_absolute_error(y_test, dec_tree.predict(X_test)).round(5))
        print("Test R2:", r2_score(y_test, dec_tree.predict(X_test)).round(5))

        Test MSE: 0.10832
        Test MAE: 0.15561
        Test R2: 0.67102
```

Hiperparámetros asociados

Mediante los hiperparámetros controlamos profundidad y fragmentación de un árbol.

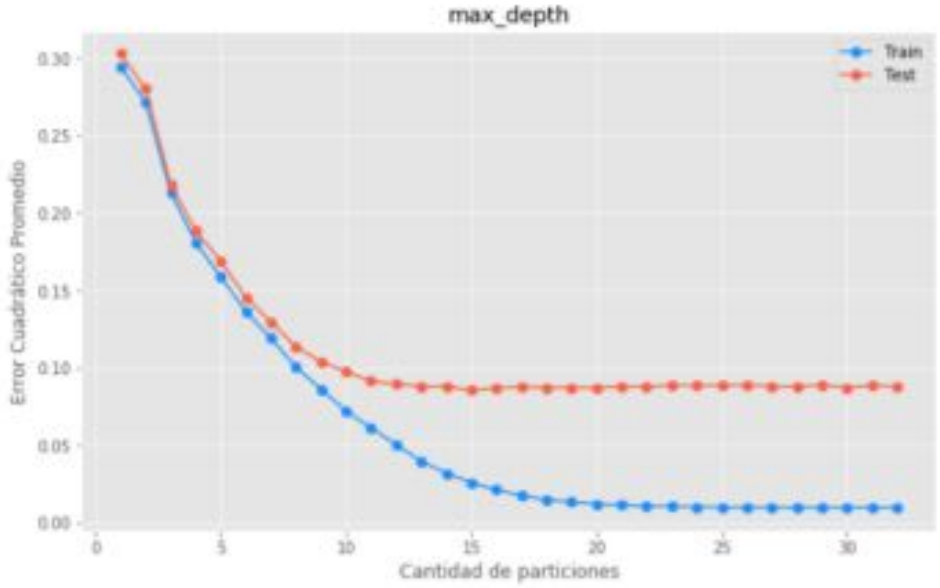
`max_depth`
Máximo de profundidad

- En la medida que dejamos que los nodos crezcan hasta alcanzar la pureza, capturamos toda la varianza del fenómeno.

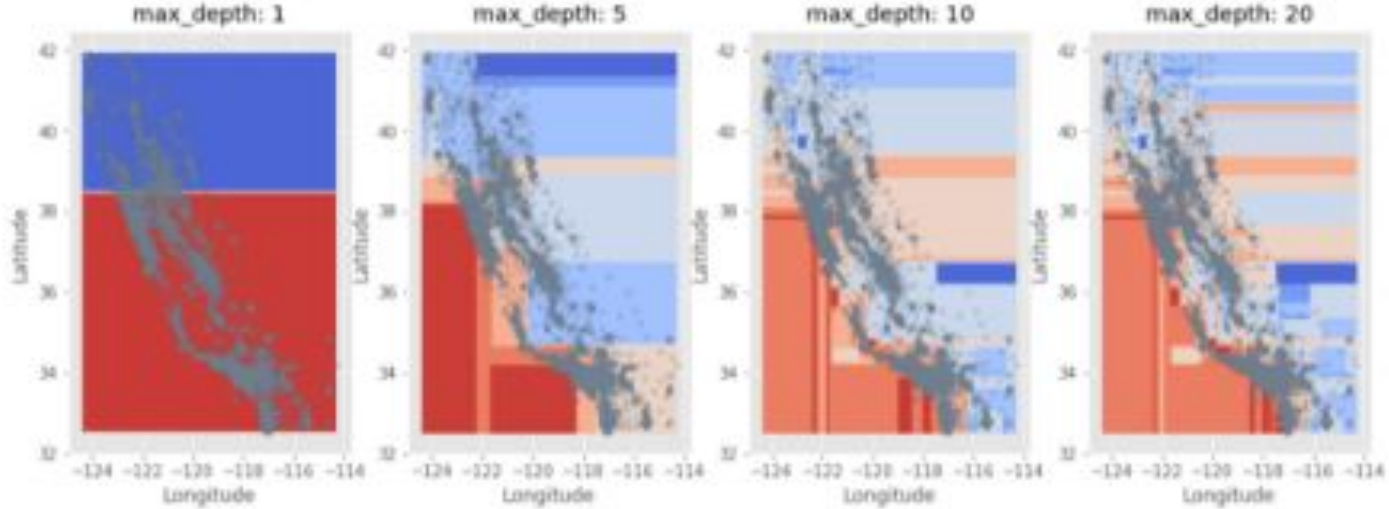
- Profundidad

  ∞: Overfitting.

```
In [11]: max_depth_train_test()
```



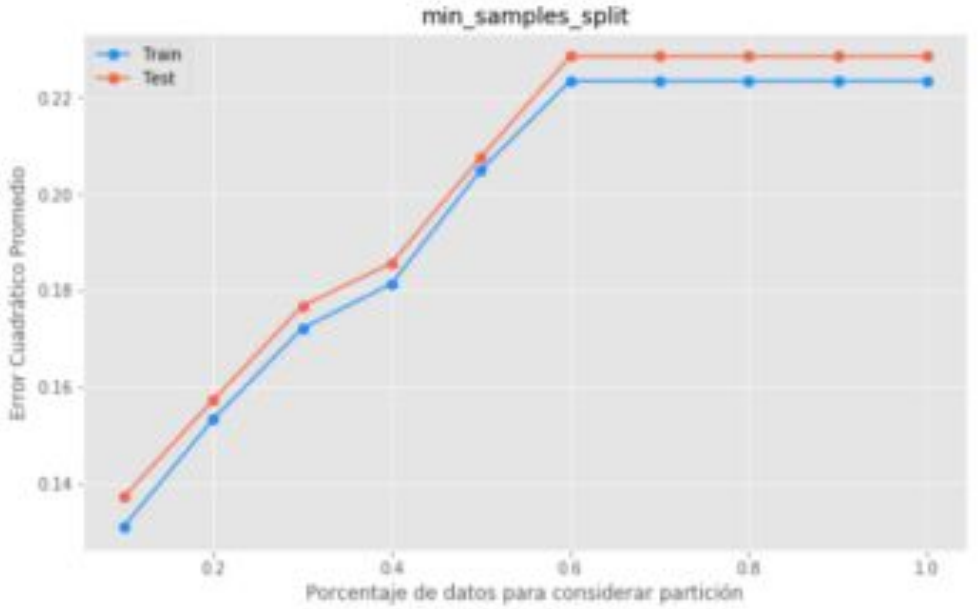
In [13]:
```
plt.figure(figsize=(15, 5)); max_depth_response_surface()
```



`min_samples_split`
Mínimo de muestras en un nodo particionable

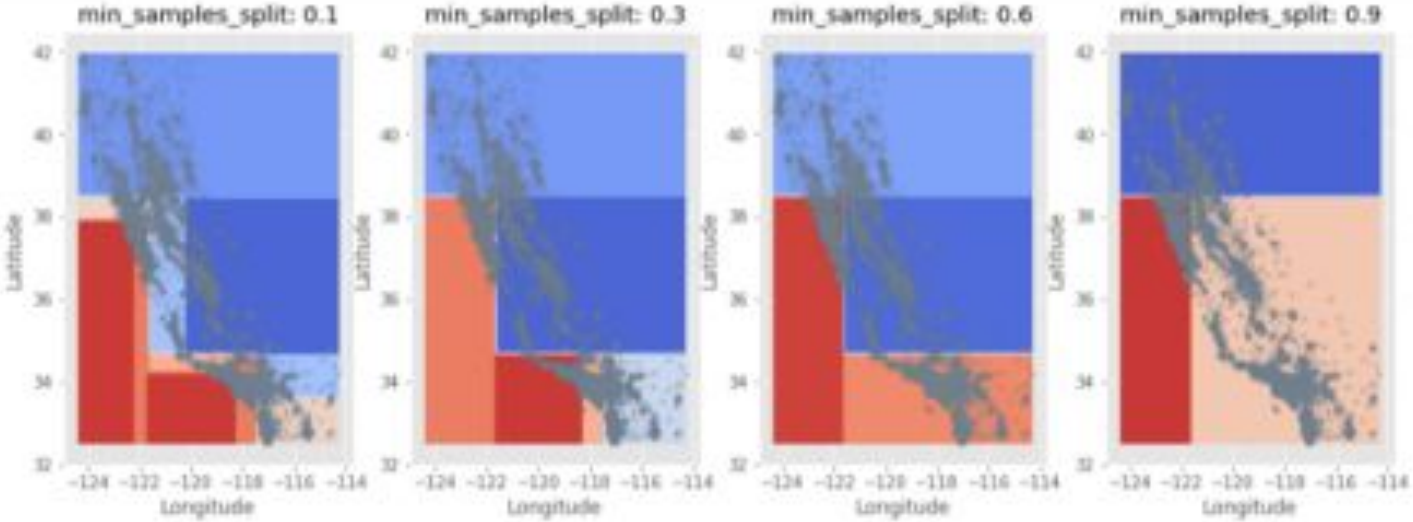
- Definimos cuándo un nodo se puede considerar como particionable, considerando la cantidad de observaciones.
- En la medida que aumenta la cantidad denobservaciones, el árbol se torna menos flexible.

```
In [15]: min_sample_train_test()
```



{desafío} latam_

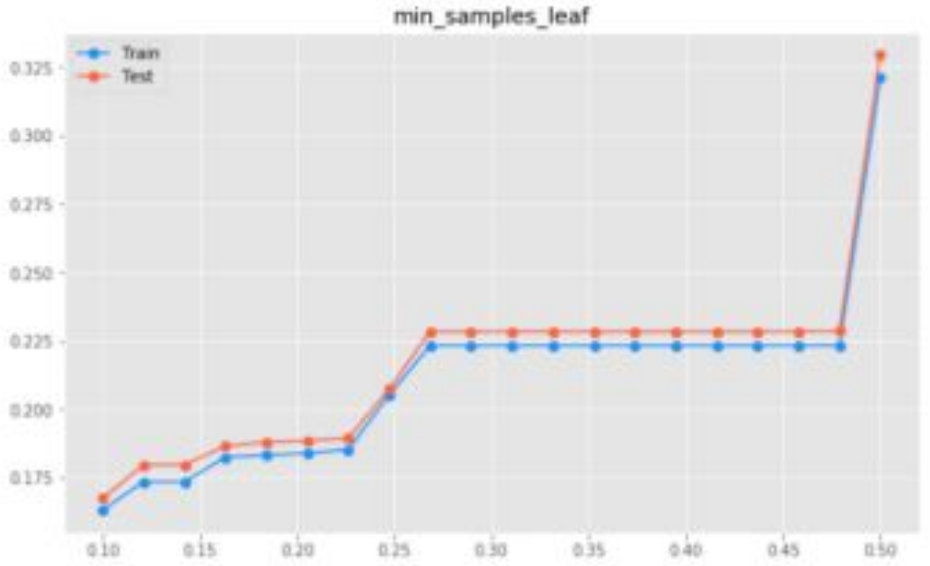
In [17]:
```
plt.figure(figsize=(15, 5)); min_sample_response_surface()
```



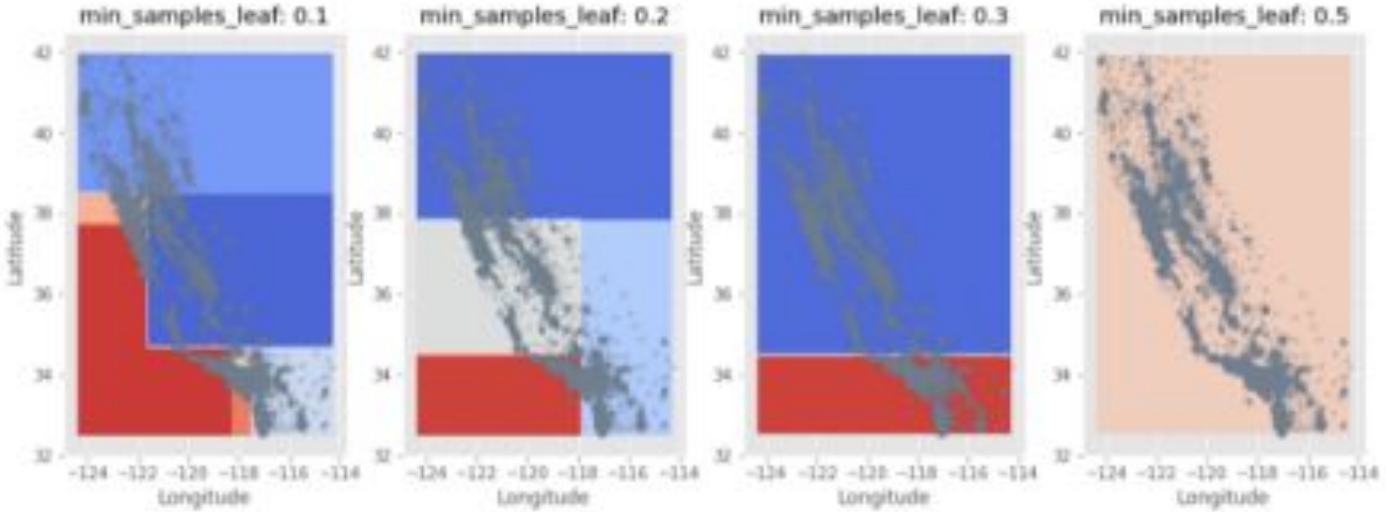
`min_samples_leaf`
Mínimo de muestras en un nodo terminal

- Definimos cuándo un nodo se puede considerar como terminal, considerando la cantidad de observaciones.
- En la medida que aumenta la cantidad de observaciones, el árbol se torna menos flexible.

```
In [19]: min_leaf_train_test()
```

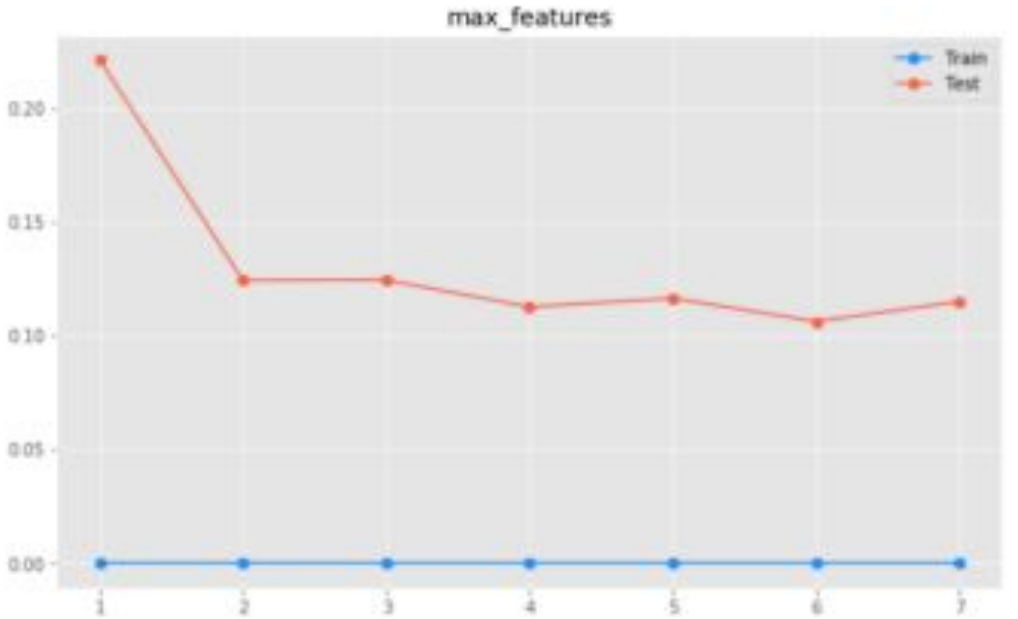


In [21]:
```
plt.figure(figsize=(15, 5)); min_leaf_response_surface()
```



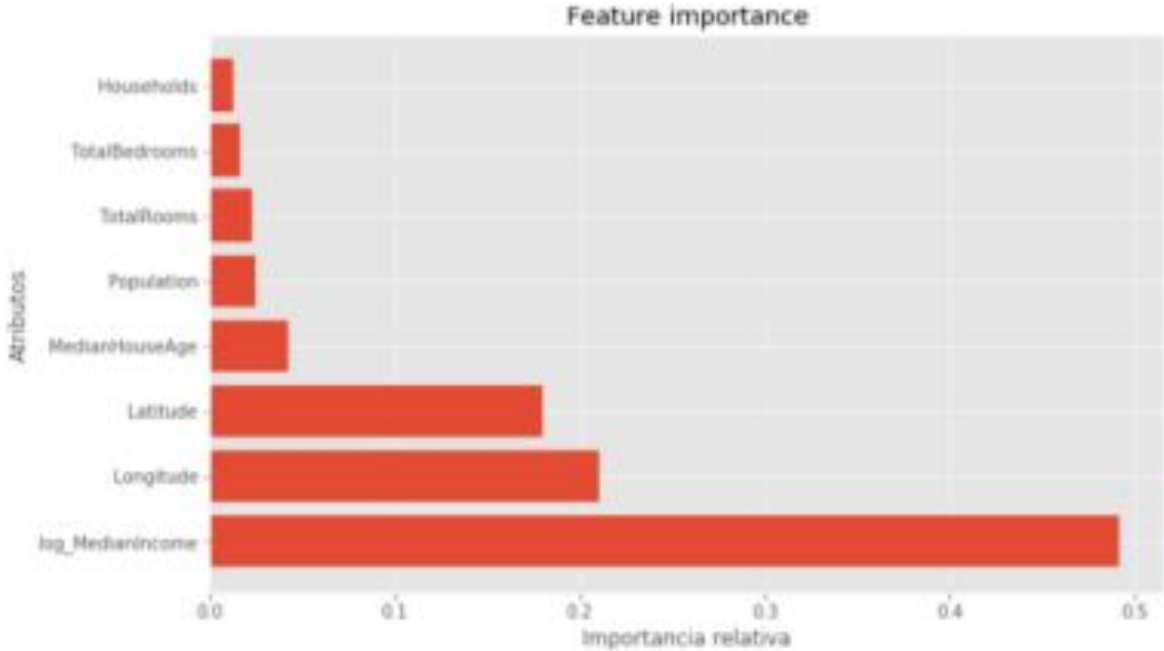
`max_features`
Cantidad de atributos

```
In [23]: max_feat_train_test()
```



- **Navaja de Occam**: Debemos siempre optar por el modelo con menor cantidad de factores explicativos.

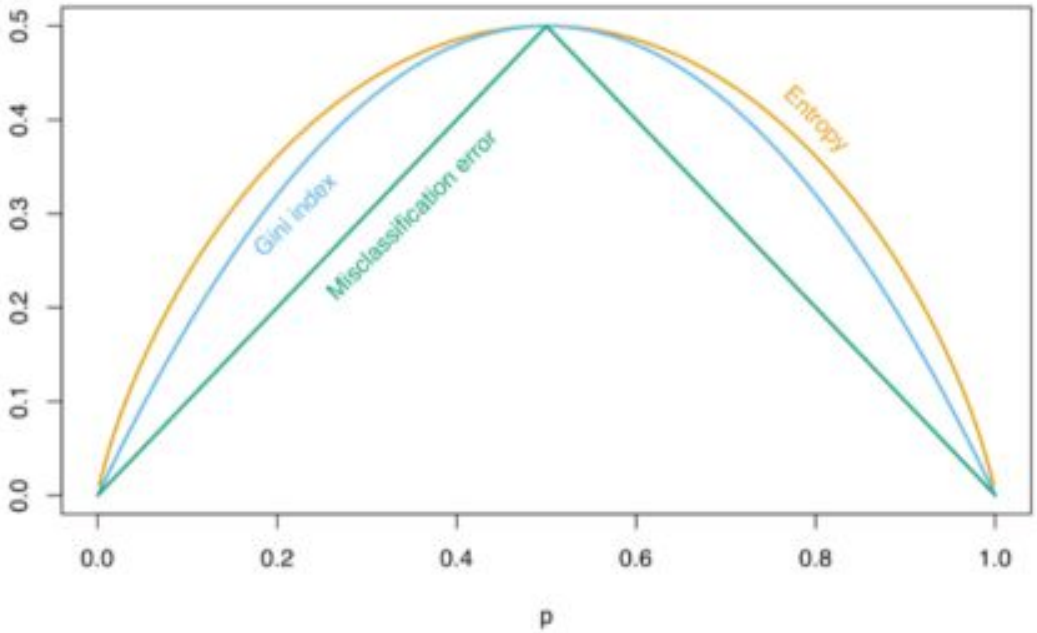
In [24]:
```
afx.plot_importance(dec_tree, X.columns);plt.xlabel("Importancia relativa");plt.ylabel('A
tributos');
```



Árboles de clasificación

Criterios de pureza

La única diferencia entre árboles de regresión y clasificación es el criterio de optimización de particiones.



Ejemplo: Identificando patrones de votación

In [25]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [26]:
```
# Importamos la base de datos
df = pd.read_csv('voting.csv').drop(columns='Unnamed: 0')
```

In [27]:
```
df.sample(1, random_state=11238)
```

Out[27]:

	region	population	sex	age	education	income	statusquo	vote
206	N	3750	M	21.0	P	15000.0	-0.04558	N

In [29]:
```
df.sample(2)
```

Out[29]:

	population	age	income	statusquo	region_M	region_N	region_S	re
1023	25000	62.0	15000.0	0.99792	0	0	1	0
607	175000	37.0	35000.0	1.43793	0	0	0	0

GridSearchCV en un árbol de clasificación

In [31]:
```
from sklearn.model_selection import GridSearchCV
```

In [32]:
```
%%time
dec_tree_grid_cv = GridSearchCV(DecisionTreeClassifier(),
                                {'min_samples_split': np.linspace(0.1, 1.0, 10),
                                 'criterion': ['gini', 'entropy'],
                                'max_depth': np.linspace(1, 32, 32),
                                'min_samples_leaf': np.linspace(0.1, 0.5, 10),
                                'max_features': list(range(1,X_train.shape[1]))},
                                 cv=5,
                                 n_jobs=-1).fit(X_train, y_train)
```

```
CPU times: user 4min 8s, sys: 4.99 s, total: 4min 13s
Wall time: 10min 40s
```

Mejores hiperparámetros

```
In [33]: dec_tree_grid_cv.best_params_

Out[33]: {'criterion': 'entropy',
          'max_depth': 3.0,
          'max_features': 8,
          'min_samples_leaf': 0.1,
          'min_samples_split': 0.1}
```

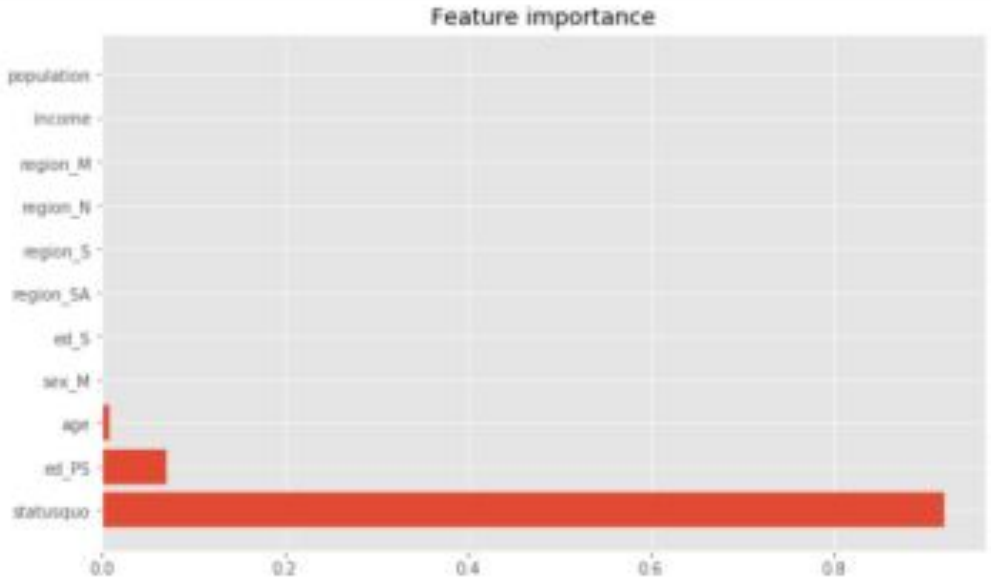
Desempeño en el training

```
In [34]: dec_tree_grid_cv.best_score_

Out[34]: 0.8357348703170029
```

Importancia de variables

```
In [35]: colnames = df.loc[:, 'population':'sex_M'].columns
         afx.plot_importance(dec_tree_grid_cv.best_estimator_, colnames)
```



Desempeño en el testing set

In [36]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test,
                           dec_tree_grid_cv.best_estimator_.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.84	0.77	0.80	572
1	0.60	0.71	0.65	283
avg / total	0.76	0.75	0.75	855

/* Desafío */

Panel de discusión

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com