```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.tree import plot_tree
import seaborn as sns
```

Primer Punto

Cargar Dataset

```
df=pd.read_csv("C:/Users/yulpr/Downloads/autos.csv")
```

	cilindrada	peso_kg	potencia_hp	consumo_km_l
0	1300	950	75	17.2
1	1500	1100	90	15.4
2	1600	1200	100	14.1
3	1800	1250	110	13.6
4	2000	1300	120	12.8

Primeras Filas e Información General

```
print("Primeras Filas:", "\n",df.head())
print("\nInformación General:","\n",df.info())
print("\nEstadísticas Descriptivas:","\n", df.describe())
```

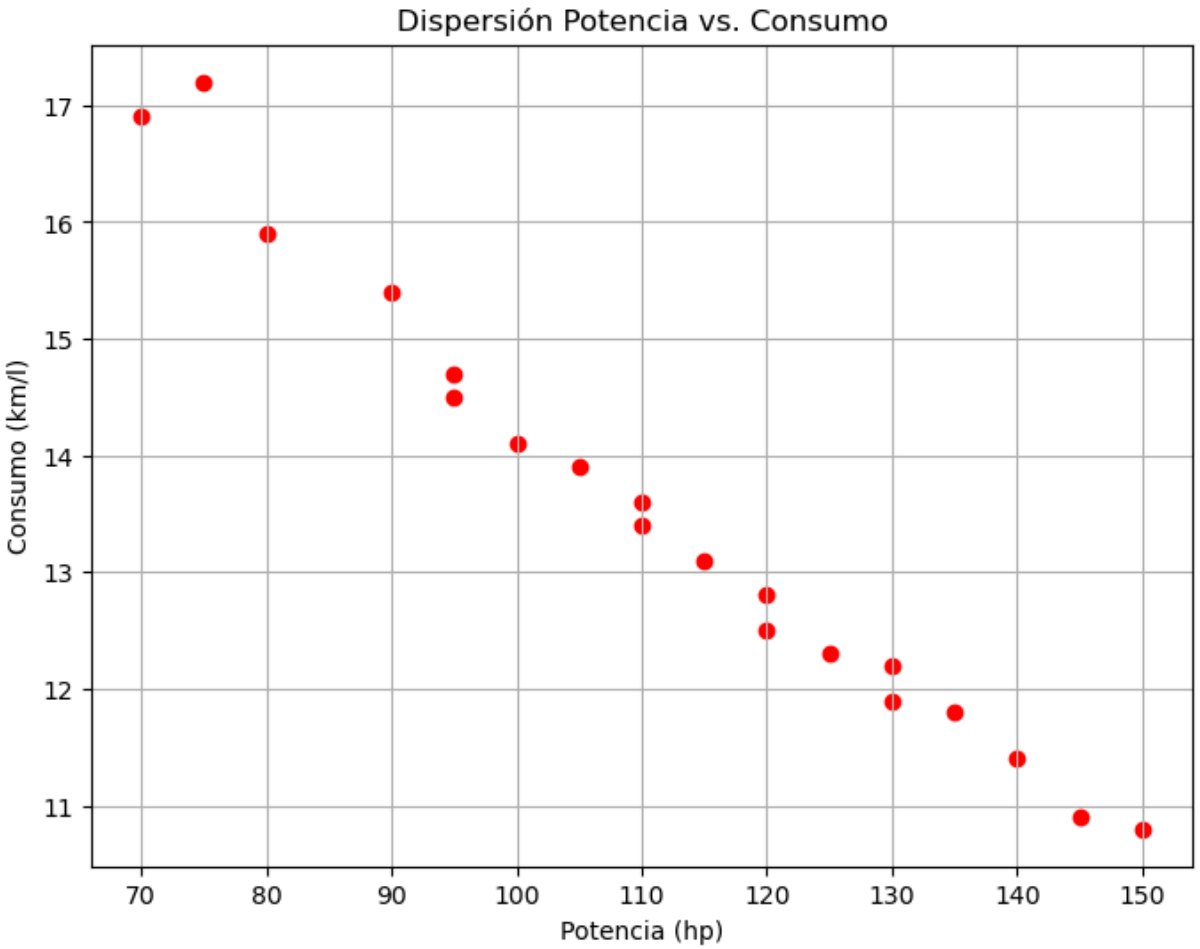
```
Primeras Filas:
    cilindrada  peso_kg  potencia_hp  consumo_km_l
0         1300      950           75          17.2
1         1500     1100           90          15.4
2         1600     1200          100          14.1
3         1800     1250          110          13.6
4         2000     1300          120          12.8
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   cilindrada    20 non-null     int64
 1   peso_kg       20 non-null     int64
 2   potencia_hp   20 non-null     int64
 3   consumo_km_l  20 non-null     float64
dtypes: float64(1), int64(3)
memory usage: 772.0 bytes

Información General:
 None

Estadísticas Descriptivas:
         cilindrada      peso_kg  potencia_hp  consumo_km_l
count    20.000000    20.000000    20.000000     20.000000
mean   1875.000000  1254.000000   112.000000     13.465000
std     386.448063   161.812107    23.192558      1.867527
min    1300.000000   950.000000    70.000000     10.800000
25%    1600.000000  1165.000000    95.000000     12.125000
50%    1800.000000  1270.000000   112.500000     13.250000
75%    2200.000000  1357.500000   130.000000     14.550000
max    2500.000000  1500.000000   150.000000     17.200000
```

Gráfico de Dispersión

```python
plt.figure(figsize=(8, 6))
plt.scatter(df['potencia_hp'], df['consumo_km_l'], color='red')
plt.title('Dispersión Potencia vs. Consumo')
plt.xlabel('Potencia (hp)')
plt.ylabel('Consumo (km/l)')
plt.grid(True)
plt.show()
```



El gráfico muestra una relación inversa ya que a medida que la potencia aumenta, el consumo disminuye, es decir, el auto consume más combustible por kilómetro recorrido.

Definir Variables Dependientes e Independientes

```
X = df[['cilindrada', 'peso_kg', 'potencia_hp']]
y = df['consumo_km_l']
```

Dividir en Entrenamiento y Prueba

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

Crear y Entrenar el Modelo

```
modelo=DecisionTreeRegressor(random_state= 42, max_depth=4)
modelo.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=4, random_state=42)
```

Predicciones Sobre el Set de Prueba

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

Evaluar Modelo

```
mse= mean_squared_error(y_test, y_pred)
r2= r2_score(y_test,y_pred)
print("mse es: ", mse)
print("r2 es : ", r2)
```

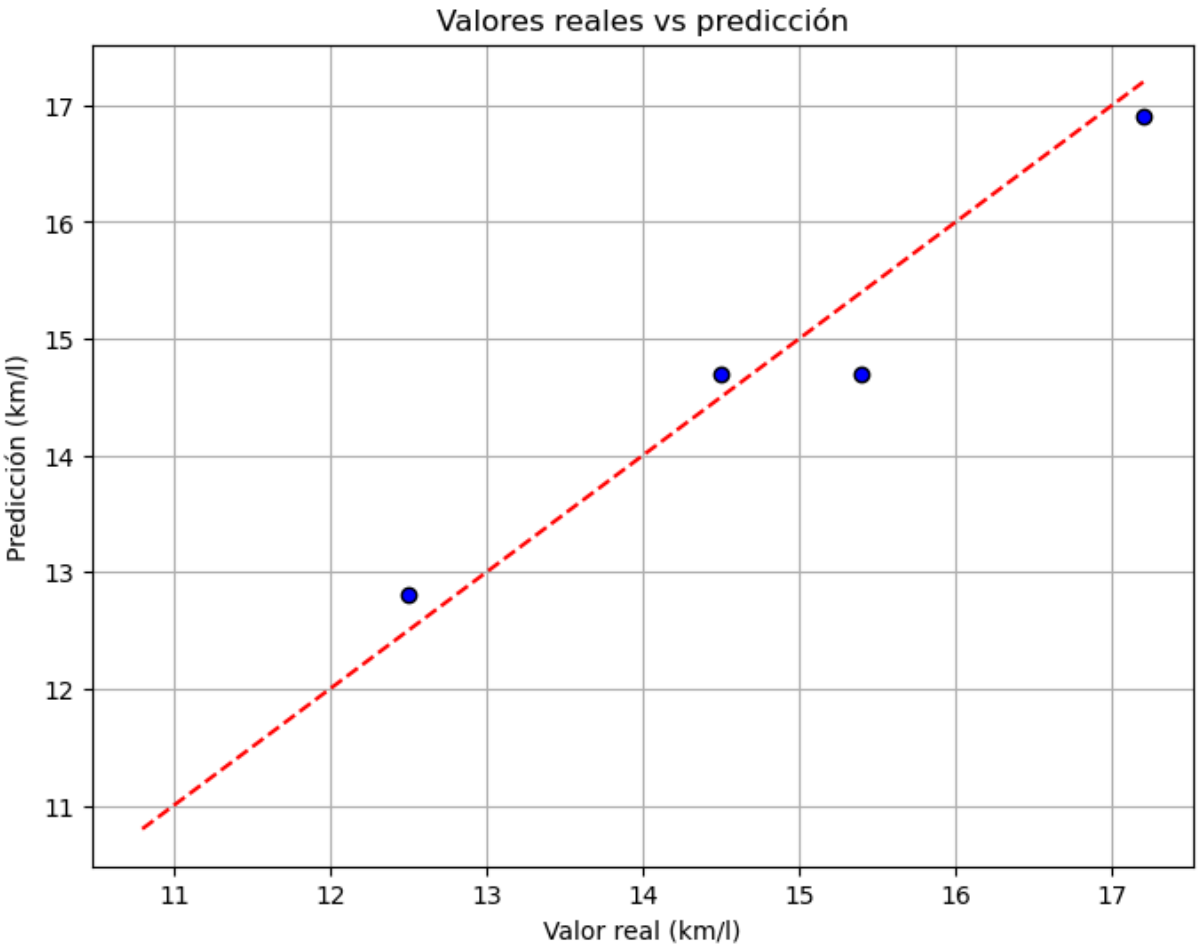
```
mse es:  0.17750000000000052
r2 es :  0.9380453752181499
```

En base a los resultados se puede decir que la predicción es bastante precisa con un margen de error de del 0.177 y un coeficiente de determinación mayor al 80%. De hecho, según entiendo el 93.8% del porqué un carro consume más o menos que otro se explica con los datos que se le asignaron al modelo.

Gráfica de Valores Reales vs predichos

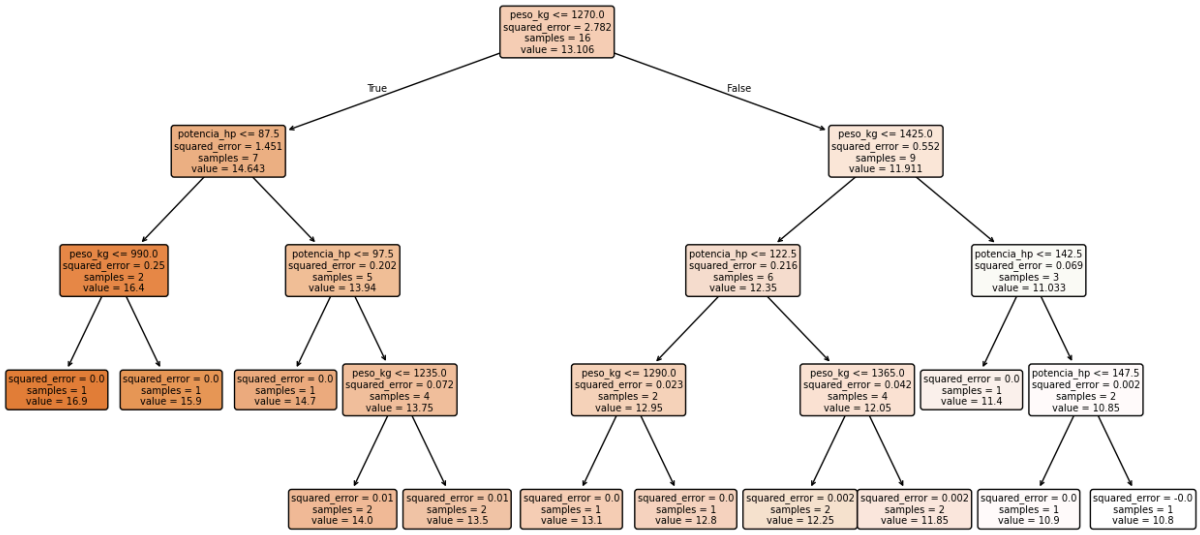
```
plt.figure(figsize=(8,6))
plt.scatter(y_test,y_pred,color='blue', edgecolor='black')
plt.plot([y.min(), y.max()],[y.min(),y.max()],'r--')
plt.title("Valores reales vs predicción")
plt.xlabel("Valor real (km/l)")
plt.ylabel("Predicción (km/l)")
plt.grid(True)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Árbol de Regresión

```
plt.figure(figsize=(16,8))
plot_tree(modelo, feature_names=X.columns, filled=True, rounded=True)
plt.show()
```



Predicción Nueva

```
nuevo_a = pd.DataFrame({
    'cilindrada': [1300],
    'peso_kg': [950],
    'potencia_hp': [75]})
prediccion= modelo.predict(nuevo_a)
print("El consumo de combustible predicho para el nuevo vehículo es:", prediccion,
```

```
El consumo de combustible predicho para el nuevo vehículo es: [16.9] km/l
```

Al ingresar la misma información de la primera fila del dataset no son iguales, pero tampoco es mucha la diferencia y teniendo en cuenta que el coeficiente de determinación es del 93,8% es claro que no daría el mismo valor.

Segundo Punto

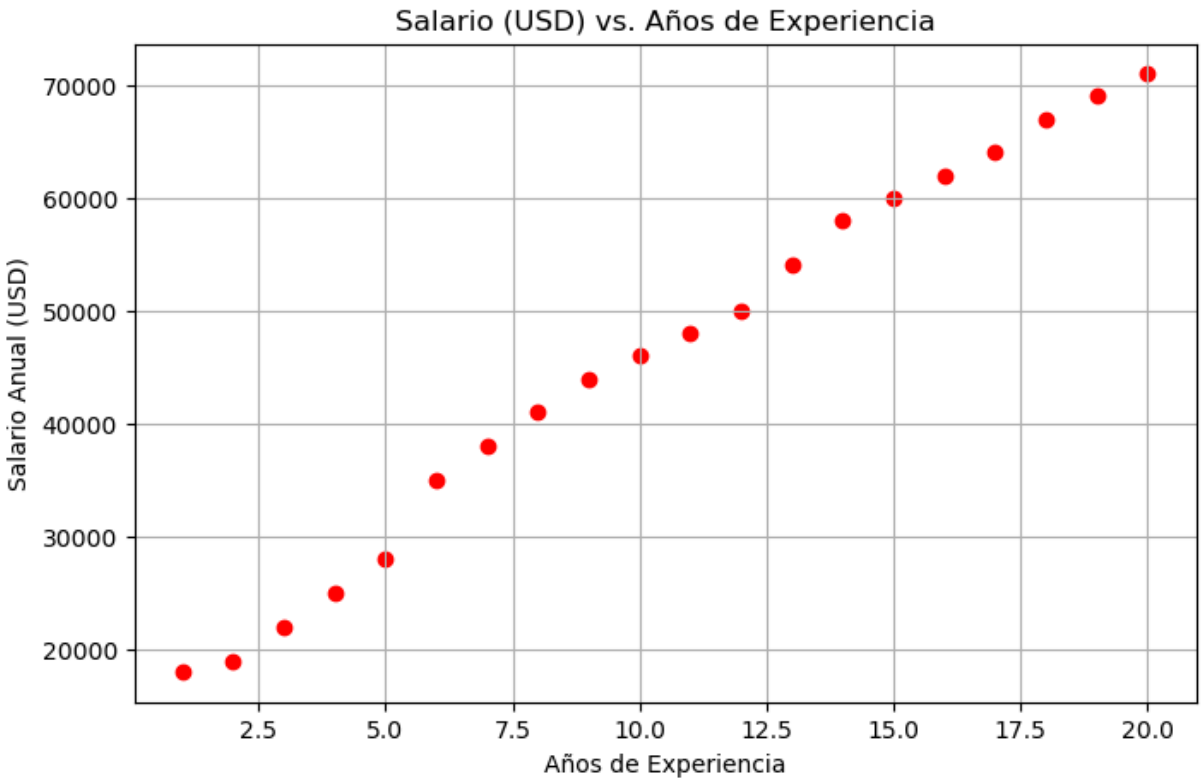
Cargar Dataset

```
df_sa=pd.read_csv("C:/Users/yulpr/Downloads/salarios.csv")
df_sa.head()
```

	experiencia_anios	nivel_educativo	horas_semana	salario_usd
0	1	1	38	18000
1	2	1	40	19000
2	3	2	42	22000
3	4	2	44	25000
4	5	2	45	28000

Relación Salario- Años de Experiencia

```
plt.figure(figsize=(8, 5))
plt.scatter(df_sa['experiencia_anios'], df_sa['salario_usd'], color='red')
plt.title('Salario (USD) vs. Años de Experiencia')
plt.xlabel('Años de Experiencia')
plt.ylabel('Salario Anual (USD)')
plt.grid(True)
plt.show()
```



En la gráfica se observa una relación positiva entre los años de experiencia y el salario. A mayor experiencia, mayor es el salario. La tendencia no es perfectamente lineal, sino que parece tener una forma escalonada, es decir que tiene un crecimiento acelerado.

Definir Variables Dependientes e Independientes

```
X=df[['experiencia_anios','nivel_educativo','horas_semana']]
y=df['salario_usd']
```

Dividir Entrenamiento y Prueba

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

Crear y Entrenar Modelo

```
modelo=DecisionTreeRegressor(random_state= 42, max_depth=5)
modelo.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5, random_state=42)
```

Predicciones

```
y_pred_train = modelo.predict(X_train)
y_pred_test = modelo.predict(X_test)
```

Evaluar Modelo

```
mse_test = mean_squared_error(y_test, y_pred_test)
r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)
print("R² en Entrenamiento:", r2_train)
print("R² en Prueba:", r2_test)
print("mse en Prueba: ", mse_test)
```

```
R² en Entrenamiento: 1.0
R² en Prueba: 0.982151244715829
mse en Prueba:  9500000.0
```

Prueba de Sobreajuste

```
if r2_train > 0.95 and r2_train - r2_test > 0.1:
    print("Posible Sobreajuste: El R² de entrenamiento es muy alto y es significati
else:
    print("El modelo muestra buen desempeño y no hay evidencia de sobreajuste.")
```

```
El modelo muestra buen desempeño y no hay evidencia de sobreajuste.
```

Importancia de Variables

```
feature_importances = pd.DataFrame({
    'Variable': X.columns,
    'Importancia': modelo.feature_importances_
}).sort_values(by='Importancia', ascending=False)
print(feature_importances.to_string(index=False, float_format='{:.3f}'.format))
```

```
        Variable  Importancia
    horas_semana        0.843
experiencia_anios        0.133
  nivel_educativo        0.024
```

La variable con mayor influencia son las horas trabajadas por semana, creo que el modelo busca la forma más rápida de clasificar a las personas en un rango de salario alto o bajo. Mirando por encima el dataset, la diferencia de salario entre un empleado que trabaja pocas horas y uno que trabaja muchas horas es notable, por eso el modelo "sabe" que mientras más horas, su salario será mayor, sin tener que revisar la experiencia o el nivel.

Predicción Nueva

```
nuevo_em = pd.DataFrame({
    'experiencia_anios': [7],
    'nivel_educativo': [3],
    'horas_semana': [45]})
prediccion_sa = modelo.predict(nuevo_em)
print("El salario anual predicho para el nuevo empleado es:",prediccion_sa, "USD")
```

```
El salario anual predicho para el nuevo empleado es: [28000.] USD
```

El modelo refleja bien la realidad de los datos del dataset. Y a pesar del alto rendimiento del modelo, el salario está influenciado por múltiples factores como la ubicación geográfica, el sector e industria, las habilidades no medibles, la oferta y demanda del mercado, el desempeño individual y la empresa como tal.

Punto 3

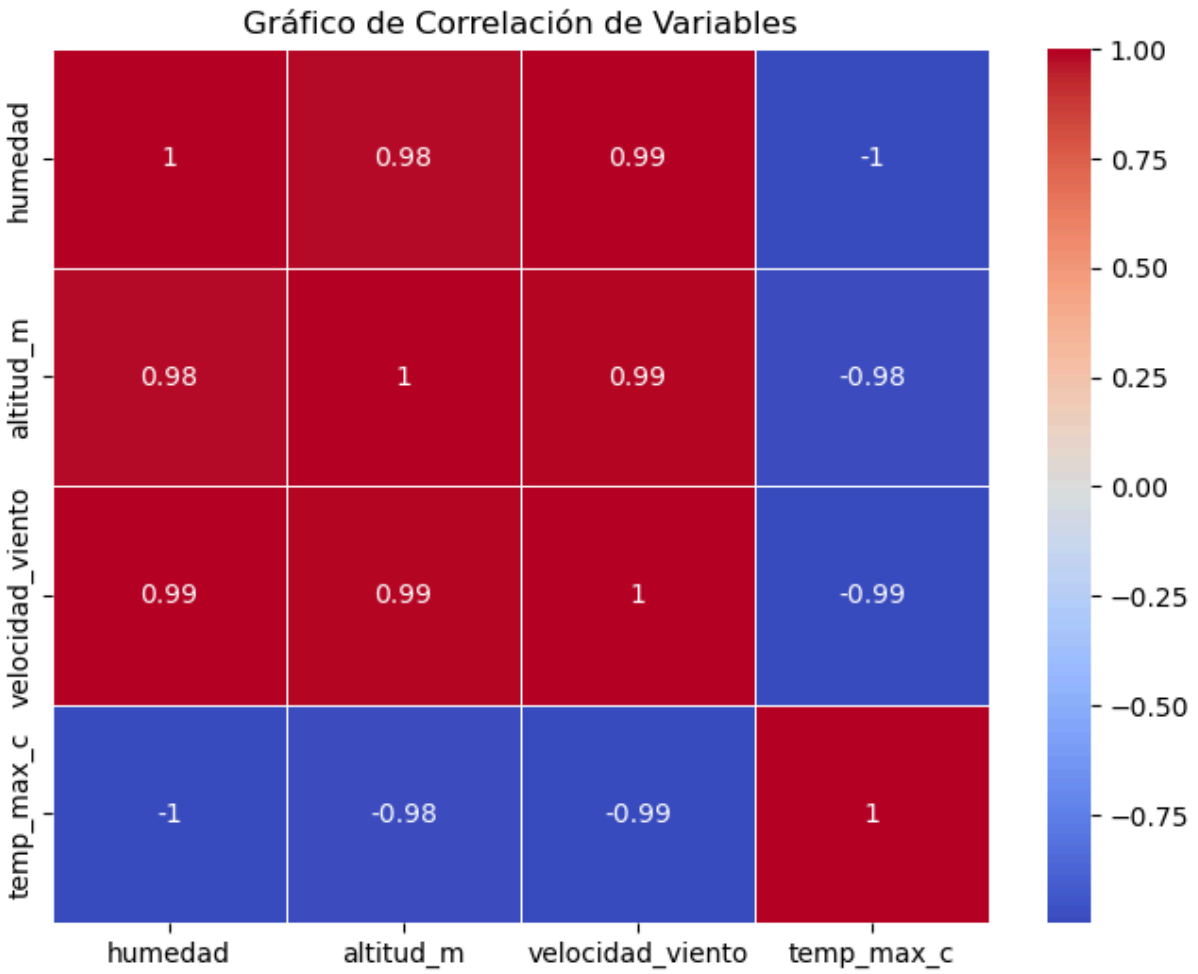
Cargar Dataset

```
df_cli=pd.read_csv("C:/Users/yulpr/Downloads/clima.csv")
df_cli.head()
```

	humedad	altitud_m	velocidad_viento	temp_max_c
0	60	100	5	32.1
1	65	150	6	31.5
2	70	200	8	30.7
3	75	300	9	29.9
4	80	400	10	29.0

Gráfico de Correlación

```
plt.figure(figsize=(8, 6))
sns.heatmap(df_cli.corr(), annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Gráfico de Correlación de Variables')
plt.show()
```



La correlación es negativa así que a medida que la altitud aumenta, la temperatura disminuye bastante.

Definir Variables

```
X = df_cli[['humedad','altitud_m','velocidad_viento']]
y = df_cli['temp_max_c']
```

Dividir en Entrenamiento y Prueba

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size= 0.2, random_sta
```

Crear y Entrenar el Modelo

```
modelo=DecisionTreeRegressor(random_state= 42, max_depth=3)
modelo.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3, random_state=42)
```

Predicción

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

```
mse= mean_squared_error(y_test, y_pred)
r2= r2_score(y_test,y_pred)
print("mse es: ", mse)
print("r2 es : ", r2)
```

```
mse es:  0.8006249999999964
r2 es :  -0.341361256544499
```

Comparación con mac_depth Mayor

```
modelo=DecisionTreeRegressor(random_state= 42, max_depth=5)
modelo.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5, random_state=42)
```

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

```
mse= mean_squared_error(y_test, y_pred)
r2= r2_score(y_test,y_pred)
print("mse es: ", mse)
print("r2 es : ", r2)
```

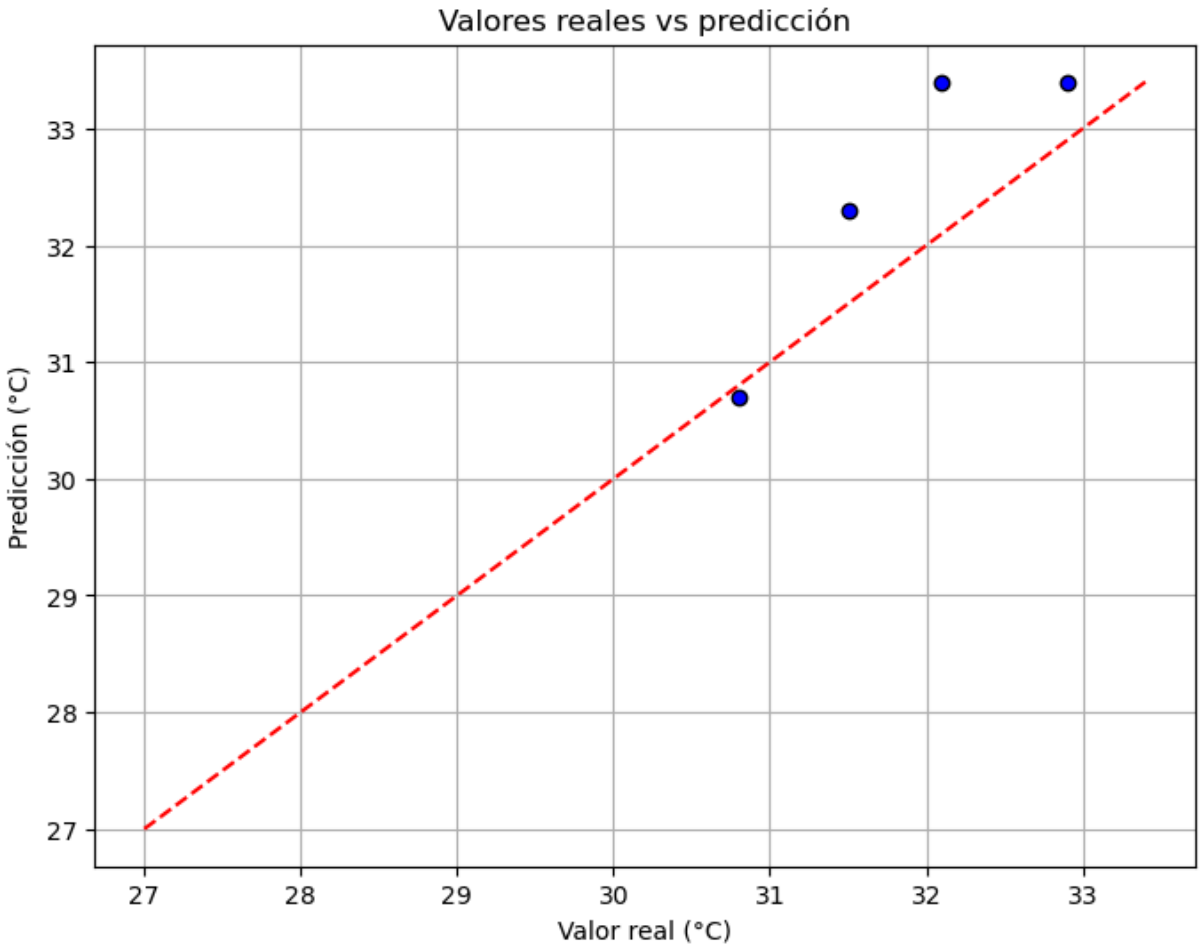
```
mse es:  0.6474999999999971
r2 es :  -0.08481675392669863
```

Con el max_depth más alto mejoró la presición, aunque r2 en ambos casos es negativo. La razón es que el dataset de prueba es pequeño, entonces no tiene suficientes ejemplos en la prueba para demostrar su precisión.

Gráfica Resultados Reales vs Predichos

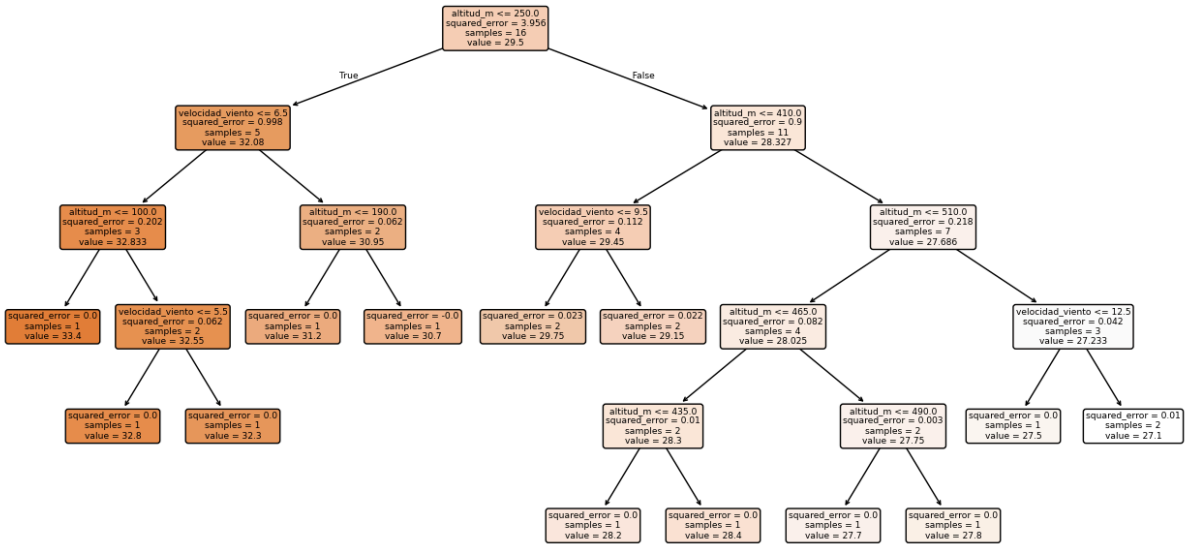
```
plt.figure(figsize=(8,6))
plt.scatter(y_test,y_pred,color='blue', edgecolor='black')
plt.plot([y.min(), y.max()],[y.min(),y.max()],'r--')
plt.title("Valores reales vs predicción")
plt.xlabel("Valor real (°C)")
plt.ylabel("Predicción (°C)")
plt.grid(True)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Visualización del Árbol

```
plt.figure(figsize=(16,8))
plot_tree(modelo, feature_names=X.columns, filled=True, rounded=True)
plt.show()
```



```
divisor_p = X.columns[modelo.tree_.feature[0]]
print("El primer divisor del árbol es la variable:", divisor_p)
```

```
El primer divisor del árbol es la variable: altitud_m
```

Como se observaba en la gráfica y se confirmó con divisor_p, el primera divisor es la altitud y tiene sentido la temperatura está directamente relacionada con la presión y la densidad del aire. A mayor altitud, el aire es más frío (se pierde calor y hay menos aire para retenerlo).

Predicción Nueva

```
nueva_tem= pd.DataFrame({
    'humedad':[70],
    'altitud_m': [500],
    'velocidad_viento':[10]})
prediccion=modelo.predict(nueva_tem)
print("La temperatura predicha es:", prediccion)
```

```
La temperatura predicha es: [27.8]
```

Pese a tener rendimientos en negativo, la temperatura predicha es coherente con el dataset ya que está en el rango esperado, pues las altitudes bajas tienen temperaturas más altas. Un caso similar al de la predicción está en la fila 4 del dataset y a pesar de tener 100m de altitud menos, el rango de temperatura se mantuvo.

Sección Final

1. ¿Cuál modelo presentó mejor rendimiento (R^2 más alto)?

El modeo con mejor rendimiento fue el de Salarios, tuvo un r^2 de casi el 98%, mientras que el de autos tuvo 93% y el de clima tuvo resultados negativos (por la condición del dataset).

2. ¿En cuál dataset fue más fácil identificar relaciones?

En el dataset de clima, de por sí la relación altitud-temperatura es un fenómeno físico conocido, no es necesario un análisis para identificarla.

3. ¿Cómo afectó el parámetro max_depth los resultados?

En el Punto 3, al comparar max_depth=3 con max_depth=5, la precisión mejoró ligeramente (de -0.34 a -0.08). Esto demuestra que el modelo necesitaba un poco de complejidad para disminuir su error, pero esto aumenta el riesgo de sobreajuste.

4. ¿Qué ventajas y limitaciones encontraste en los árboles de decisión para regresión?

Ventajas: 1. Muestran la ruta de decisión que tomó el modelo para llegar al resultado. 2. Son muy buenos para predecir cosas que no siguen una línea recta como los aumentos salariales ligados al cambio de nivel educativo. 3. Dicen cuál fue la variable más importante al momento de tomar una decisión.

Desventajas: 1. Si tienen un sobreajuste se aprenden los datos de memoria y fallan cuando se les da un dato nuevo. 2. Les cuesta mucho predecir valores que están muy por fuera del rango de datos que vieron. 3. Como en el Ejercicio 3, si tienen pocos ejemplos para la prueba, el cálculo de su precisión puede salir totalmente mal.