

SELECCIÓN DE CARACTERÍSTICAS

✓ 1. Introducción

En resumen, feature selection, también llamado como selección de mejores características o atributos, es el proceso de seleccionar un subconjunto de características pertinentes (variables, predictores) para su uso en construcción de modelos. Las técnicas de feature selection son utilizadas por cuatro razones:

- Simplificación de modelos con el fin de hacerlas más sencillas de interpretar para los usuarios/investigadores.
- Menor tiempo de entrenamiento.
- Evitar la maldición de la dimensionalidad:
- Generalización realzada por reducir *overfitting* (formalmente, reducción de varianza).

Todo es una ecuación 

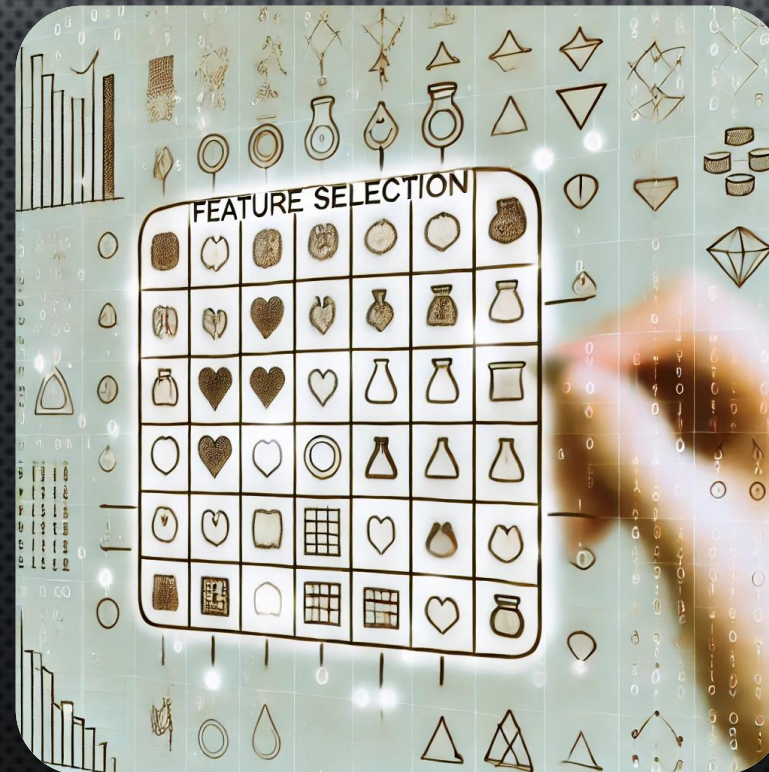
Si comprendemos que el número de características está directamente relacionado con la complejidad de la ecuación que generamos en una red neuronal, un árbol de decisión o una regresión, entonces es fundamental reconocer que no siempre es beneficioso aumentar esa complejidad

ELIMINAR ALTA CORRELACIÓN.

ELIMINAR SOBRE DIMENSIONALIDAD ==> COSTO COMPUTACIONAL Y POR OVERFITTING

MEJORAR TIEMPOS DE ENTRENAMIENTOS.

GENERAR MODELOS INTERPRETABLES PARA LAS PERSONAS

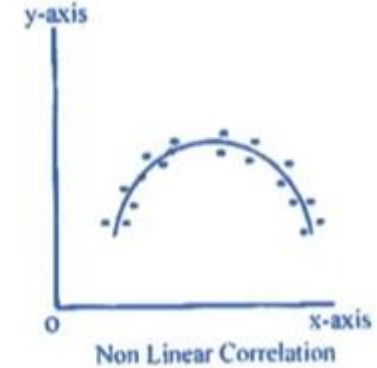
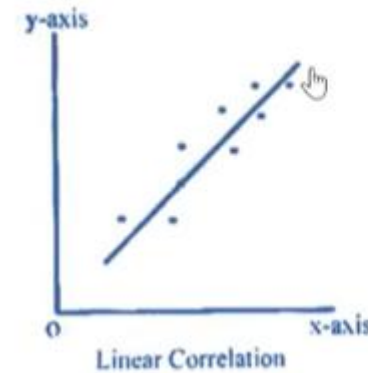


Información Mutua : Qué tanto de la información de una variable es explicable por la información de la otra variable
Eso está basada en la teoría de la información y en entropía (uno por uno entre las características)

✓ from scipy.stats import spearmanr, kendalltau ==> CORRELACIÓN NO LINEAL

import statsmodels.formula.api as smf ==> REGRESIONES LINEALES CON DATOS ESTADÍSTICOS AVANZADOS

```
[ ] # Importar las librerías necesarias
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import spearmanr, kendalltau ## NO LINEALES
from sklearn.feature_selection import mutual_info_regression, mutual_info_classif
from sklearn.datasets import fetch_california_housing, load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
import statsmodels.api as sm
import statsmodels.formula.api as smf
```




```
# Crear y entrenar el modelo
modelo_sm = sm.OLS(y_train_california, X_train_california_sm)
resultados = modelo_sm.fit()
```



Statsmodels

<https://www.statsmodels.org> › [ols](#) · [Traducir esta página](#) ⋮

✓ Ordinary Least Squares - statsmodels 0.15.0 (+415)

Draw a plot to compare the true relationship to **OLS** predictions. Confidence interval predictions are built using the `wls_prediction_std` command.

[OLS estimation](#) · [OLS non-linear curve but...](#) · [OLS with dummy variables](#)

California Housing Dataset

¿Para qué sirve este dataset?

Este tipo de dataset es utilizado para **predecir el precio de las viviendas** o **el valor medio de las propiedades** en diferentes áreas, basado en múltiples características como:

1. **MedInc (Median Income):** Representa el nivel de ingresos en la región, que generalmente está correlacionado con el precio de las propiedades.
2. **HouseAge (Edad de la Casa):** La edad promedio de las viviendas en un área puede influir en su precio.
3. **AveRooms y AveBedrms (Promedio de Habitaciones y Dormitorios):** Cuantas más habitaciones y dormitorios tiene una casa, típicamente mayor es su valor.
4. **Population (Población):** El tamaño de la población en un área puede influir en la demanda de viviendas, y por lo tanto, en su precio.
5. **AveOccup (Ocupación Promedio):** El número promedio de ocupantes por vivienda podría relacionarse con la densidad de población y la demanda en el área.
6. **Latitude y Longitude (Latitud y Longitud):** Estas variables geográficas permiten ubicar las propiedades en un mapa, lo que es esencial para capturar el efecto de la ubicación en los precios de las viviendas.



P-value = métrica que permite estimar la probabilidad de que los resultados sean debidos al azar y determinar si los hallazgos de un experimento son estadísticamente significativos. $P < 0.05$.

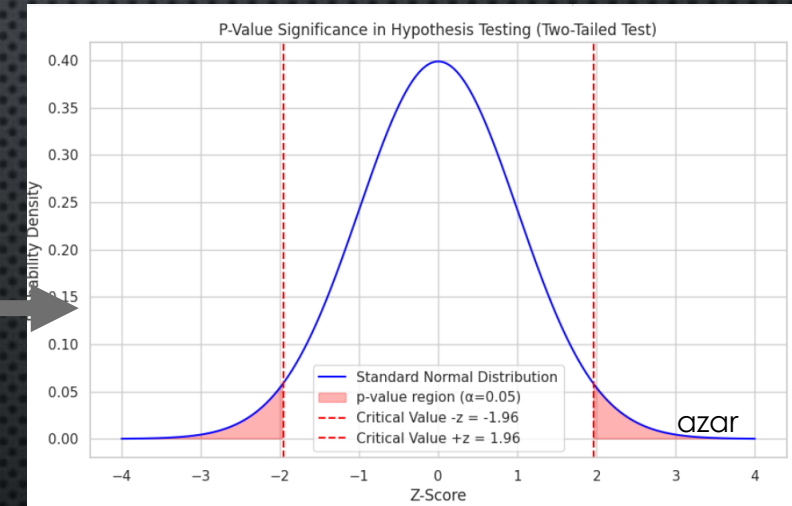
Si el p-value es mayor a 0.05 ==> resultados se deban a ruido o simplemente al azar.

```
Dep. Variable:          target    R-squared:          0.613
Model:                  OLS       Adj. R-squared:     0.612
Method:                 Least Squares   F-statistic:       3261.
Date:                   Fri, 30 Aug 2024   Prob (F-statistic): 0.00
Time:                   22:18:18         Log-Likelihood:    -17998.
No. Observations:       16512          AIC:               3.601e+04
Df Residuals:           16503          BIC:               3.608e+04
Df Model:                8
Covariance Type:        nonrobust
```

	coef	std err	t	P> t	[0.025	0.975]
const	-37.0233	0.728	-50.835	0.000	-38.451	-35.596
MedInc	0.4487	0.005	95.697	0.000	0.439	0.458
HouseAge	0.0097	0.000	19.665	0.000	0.009	0.011
AveRooms	-0.1233	0.007	-18.677	0.000	-0.136	-0.110
AveBedrms	0.7831	0.033	23.556	0.000	0.718	0.848
Population	-2.03e-06	5.25e-06	-0.387	0.699	-1.23e-05	8.26e-06
AveOccup	-0.0035	0.000	-7.253	0.000	-0.004	-0.003
Latitude	-0.4198	0.008	-52.767	0.000	-0.435	-0.404
Longitude	-0.4337	0.008	-52.117	0.000	-0.450	-0.417

```
Omnibus:          3333.187    Durbin-Watson:          1.962
Prob(Omnibus):    0.000      Jarque-Bera (JB):       9371.466
Skew:             1.071      Prob(JB):               0.00
Kurtosis:         6.006      Cond. No.               2.38e+05
```

Significa 61% de varianza de la variable objetivo es explicada por la varianza de las variables del data SET.



Cuál es la probabilidad de que los resultados que yo estoy viendo sean reales O sean atribuibles al azar.

Con que probabilidad los datos son dependientes del azar

OLS Regression Results

```

=====
Dep. Variable:          target    R-squared:          0.613
Model:                  OLS       Adj. R-squared:     0.612
Method:                 Least Squares   F-statistic:       3261.
Date:                   Fri, 30 Aug 2024   Prob (F-statistic): 0.00
Time:                   22:18:18    Log-Likelihood:    -17998.
No. Observations:      16512      AIC:               3.601e+04
Df Residuals:          16503      BIC:               3.608e+04
Df Model:               8
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	-37.0233	0.728	-50.835	0.000	-38.451	-35.596
MedInc	0.4487	0.005	95.697	0.000	0.439	0.458
HouseAge	0.0097	0.000	19.665	0.000	0.009	0.011
AveRooms	-0.1233	0.007	-18.677	0.000	-0.136	-0.110
AveBedrms	0.7831	0.033	23.556	0.000	0.718	0.848
Population	-2.03e-06	5.25e-06	-0.387	0.699	-1.23e-05	8.26e-06
AveOccup	0.0035	0.000	7.253	0.000	0.004	0.003
Latitude	-0.4198	0.008	-52.767	0.000	-0.435	-0.404
Longitude	-0.4337	0.008	-52.117	0.000	-0.450	-0.417

```

=====
Notes:
Omnibus:          3333. [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Prob(Omnibus):    0. [2] The condition number is large, 2.38e+05. This might indicate that there are
Skew:            1. strong multicollinearity or other numerical problems.
Kurtosis:        6.

6. Métricas de evaluación para la Regresión Lineal con Statsmodels:
MSE: 0.5559
MAE: 0.5332
R^2: 0.6126

```

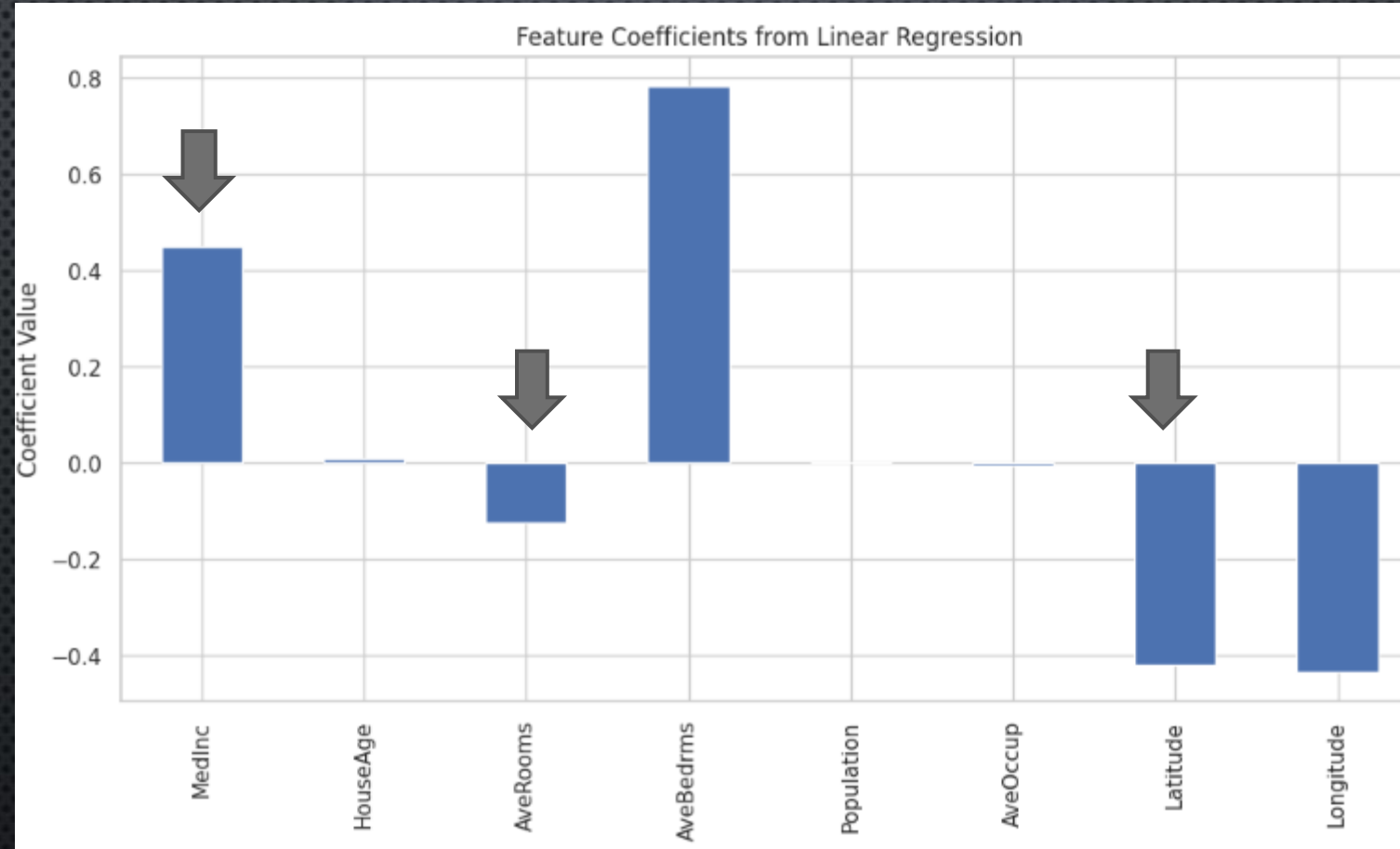
1. Los coeficientes sean altos + o -
2. P sea bajo (dataset ideal)
3. Intervalo de confianza
Si tengo un intervalo demasiado grande puedo sospechar que algo esta mal, se deba al azar

Dataset mas grande


Entre mas habitaciones son mas caras (confiable)

No afecta al modelo y no es confiable
Tiene un peso pequeño podría eliminarlo

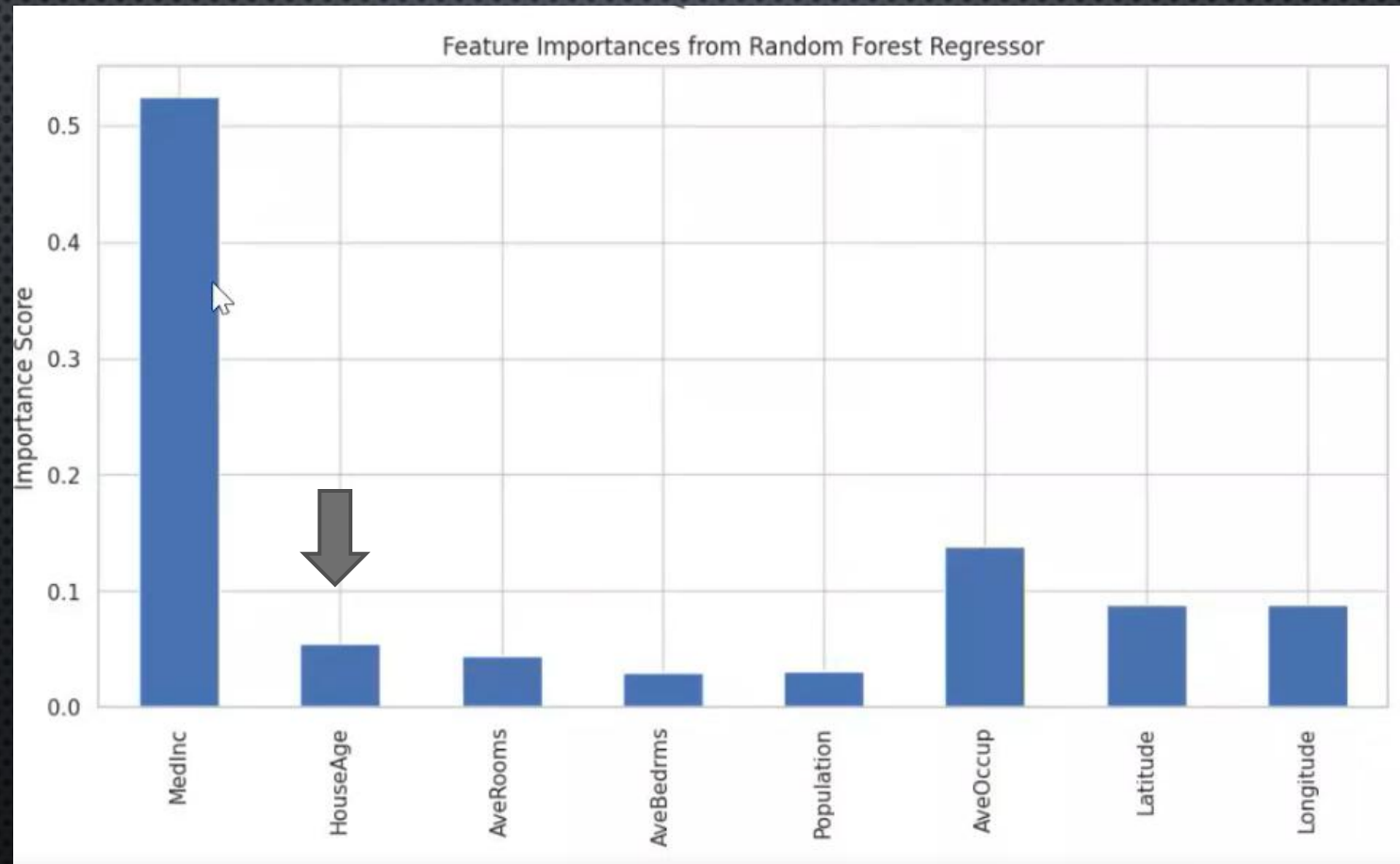
azar




python

 Copiar código

```
from sklearn.ensemble import RandomForestRegressor
```



python

 Copiar código

```
from xgboost import plot_importance
import matplotlib.pyplot as plt

# Graficar las importancias de las características
plot_importance(model)
plt.show()
```

Esta función `plot_importance` muestra un gráfico de barras con las importancias de cada característica, lo que facilita la interpretación visual.

importancias de cada característica, lo que facilita la interpretación visual.

Esta función `plot_importance` muestra un gráfico de barras con las

Métodos de Importancia en `XGBoost`


`XGBoost` ofrece diferentes formas de calcular la importancia de las características:

- **Weight:** El número de veces que una característica es usada para dividir los datos en los árboles.
- **Gain:** El aumento promedio en la precisión de la predicción que se obtiene al dividir los datos basándose en esa característica.
- **Cover:** La fracción de observaciones a las que se aplica una característica específica.

Puedes especificar el tipo de importancia que deseas visualizar usando el parámetro

`importance_type` en `plot_importance`:

python

 Copiar código

```
plot_importance(model, importance_type='gain') # Puede ser 'weight', 'gain', o 'cover'
```

```
plot_importance(model, importance_type='gain') # Puede ser 'weight', 'gain', o 'cover'
```


Sesgo hacia características de alta cardinalidad:

Problema: Los métodos como el `Gain` y `Weight` tienden a dar mayor importancia a características que tienen más niveles o valores posibles, como variables categóricas de alta cardinalidad.

Consecuencia: Esto puede resultar en una sobreestimación de la importancia de estas características, simplemente porque proporcionan más opciones para dividir los datos, no necesariamente porque sean realmente más informativas.

Dependencia de la escala de los datos:

Problema: Las características numéricas que no están escaladas pueden influir desproporcionadamente en la importancia calculada. Por ejemplo, características con mayores magnitudes (como ingresos en miles vs. gastos en unidades) pueden recibir mayor importancia simplemente debido a su escala.

Consecuencia: Esto puede llevar a interpretaciones incorrectas sobre qué características son realmente importantes.

3. Interacciones no capturadas:

- **Problema:** Los métodos de importancia de características generalmente miden la importancia de cada característica de forma independiente, sin considerar las interacciones complejas entre múltiples características.
- **Consecuencia:** Las interacciones entre características, que pueden ser cruciales para el rendimiento del modelo, no se reflejan en la importancia de las características. Esto podría llevar a subestimar la importancia de características que son relevantes solo cuando se consideran en combinación con otras.

4. Sobresimplificación:

- **Problema:** La importancia de características a menudo se basa en métricas relativamente simples, como la frecuencia de uso en divisiones ("weight") o el aumento en la precisión ("gain"), que pueden no capturar completamente la complejidad de cómo una característica impacta las predicciones.
- **Consecuencia:** Esto puede llevar a una visión simplificada y potencialmente engañosa de la importancia real de las características en el modelo.

5. Inestabilidad en la selección de características:

- **Problema:** Los métodos como los árboles de decisión en "RandomForest" y "XGBoost" pueden ser inestables, es decir, pequeñas variaciones en los datos de entrenamiento pueden llevar a grandes cambios en las características seleccionadas como importantes.
- **Consecuencia:** Esto puede hacer que la importancia de las características sea inconsistente y dependiente del subconjunto específico de datos utilizado para entrenar el modelo.

6. Sesgo hacia características más cerca de la raíz:

- **Problema:** En los árboles de decisión, las características que se utilizan para las divisiones más cerca de la raíz tienden a recibir una importancia desproporcionada, ya que afectan a un mayor número de observaciones.
- **Consecuencia:** Esto puede dar una impresión errónea de que estas características son más importantes de lo que realmente son.

7. No captura la importancia en modelos lineales:

- **Problema:** En modelos lineales, la importancia de las características a menudo se mide por los coeficientes asociados a cada característica. Sin embargo, estos coeficientes no capturan la importancia de las características que podrían ser útiles en interacciones no lineales.
- **Consecuencia:** Características que son útiles sólo en combinación con otras pueden tener coeficientes bajos, y por lo tanto ser subestimadas en términos de su importancia.

8. Problemas con multicolinealidad:

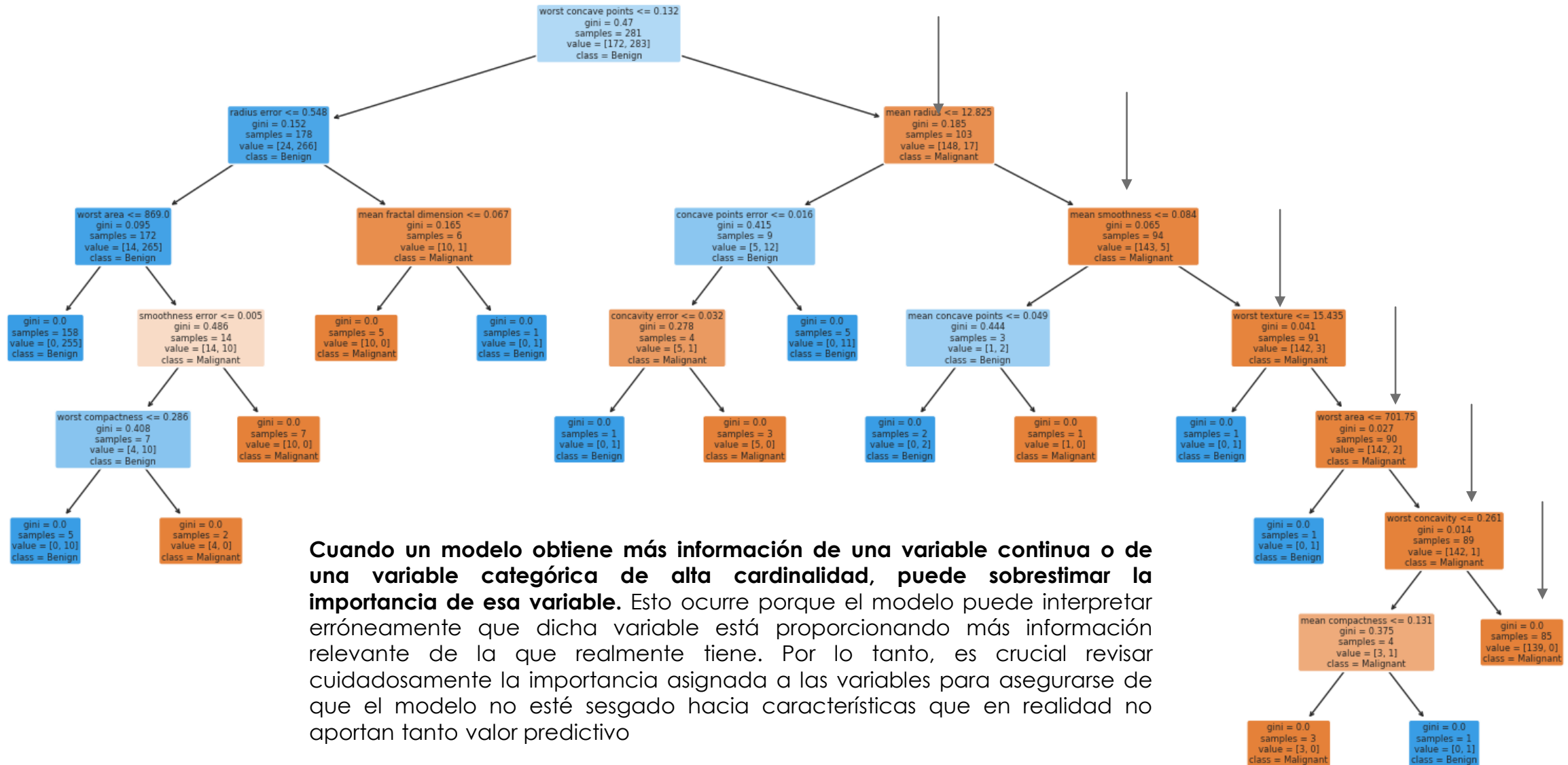
- **Problema:** Si dos o más características están altamente correlacionadas (multicolinealidad), la importancia puede ser distribuida arbitrariamente entre ellas, lo que podría resultar en una subestimación de la importancia de características que son efectivamente redundantes pero importantes.
- **Consecuencia:** Esto puede dificultar la interpretación de los resultados y llevar a decisiones incorrectas al seleccionar características.

Estos métodos de cálculo de la importancia de las características tienen un sesgo inherente, ya que tienden a sobreestimar la importancia de las variables numéricas continuas y de las variables categóricas con alta cardinalidad.

Por ejemplo, una variable categórica de alta cardinalidad es aquella que tiene cuatro o más clases, como el índice de masa corporal (IMC), que se clasifica en seis categorías: bajo peso, peso normal, sobrepeso, obesidad grado 1, obesidad grado 2 y obesidad grado 3. Estos métodos pueden sobreestimar el peso de tales variables, lo que puede llevar a la impresión equivocada de que una variable tiene mucha más influencia en el modelo de lo que realmente tiene


Opciones :

Árbol de Decisión en Random Forest



Cuando un modelo obtiene más información de una variable continua o de una variable categórica de alta cardinalidad, puede sobrestimar la importancia de esa variable. Esto ocurre porque el modelo puede interpretar erróneamente que dicha variable está proporcionando más información relevante de la que realmente tiene. Por lo tanto, es crucial revisar cuidadosamente la importancia asignada a las variables para asegurarse de que el modelo no esté sesgado hacia características que en realidad no aportan tanto valor predictivo

python

 Copiar código

```
from treeinterpreter import treeinterpreter as ti
```

```
RANDOMFOREST
```

Explicación

- `treeinterpreter`: Es una biblioteca en Python que permite desglosar las predicciones de modelos de árboles de decisión en contribuciones de las características. Esto es útil para entender cómo cada característica influye en una predicción particular.

```
# Utilizar Treeinterpreter para descomponer la predicción
instance = X_test_cla[0:1] # Ejemplo con la primera muestra del conjunto de prueba
prediction, bias, contributions = ti.predict(rf_clf, instance)
```

Modelo

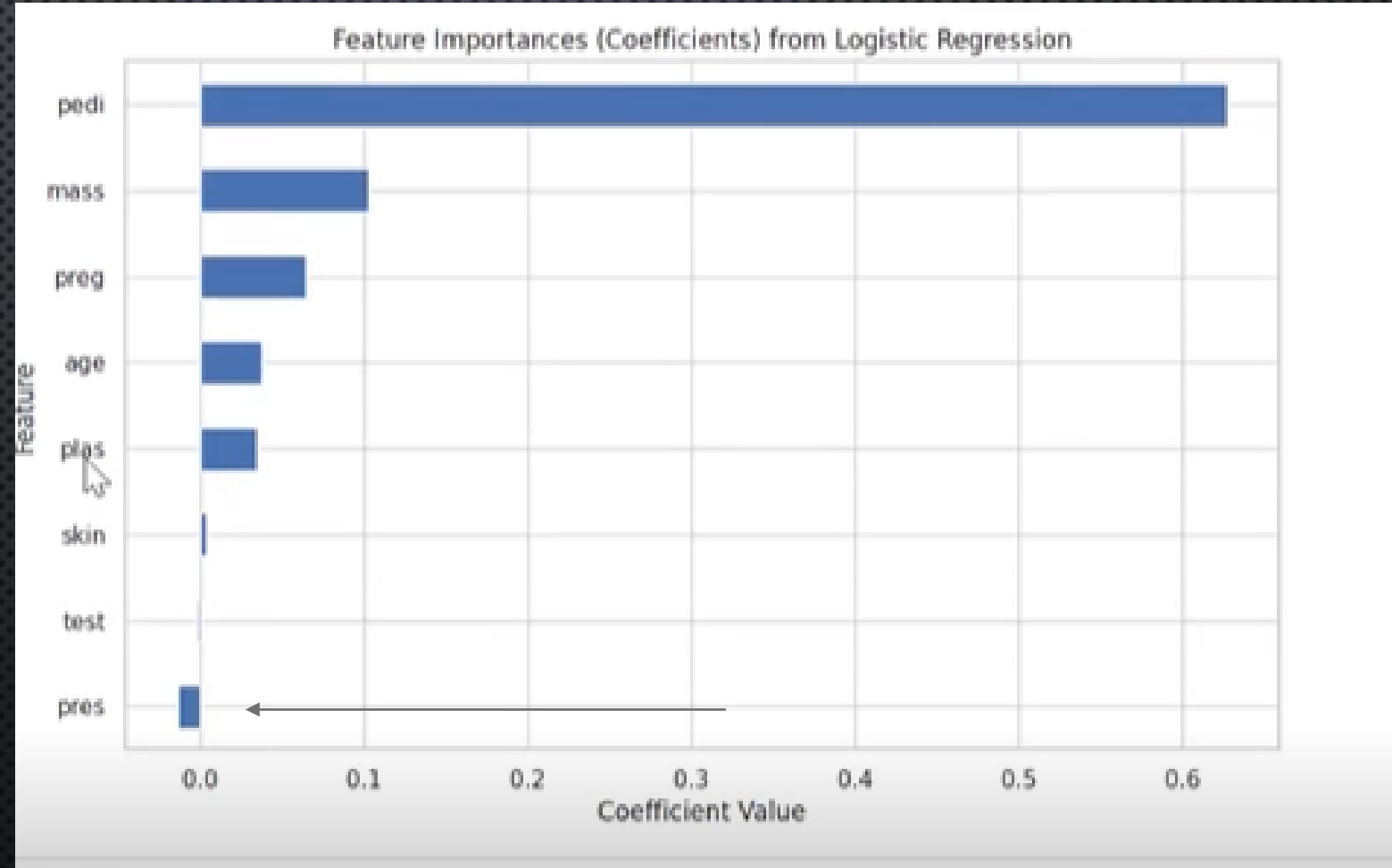
El sesgo

Y la contribución de cada variable

Cambiamos de dataset , vamos a predecir si una persona puede tener diabetes o no

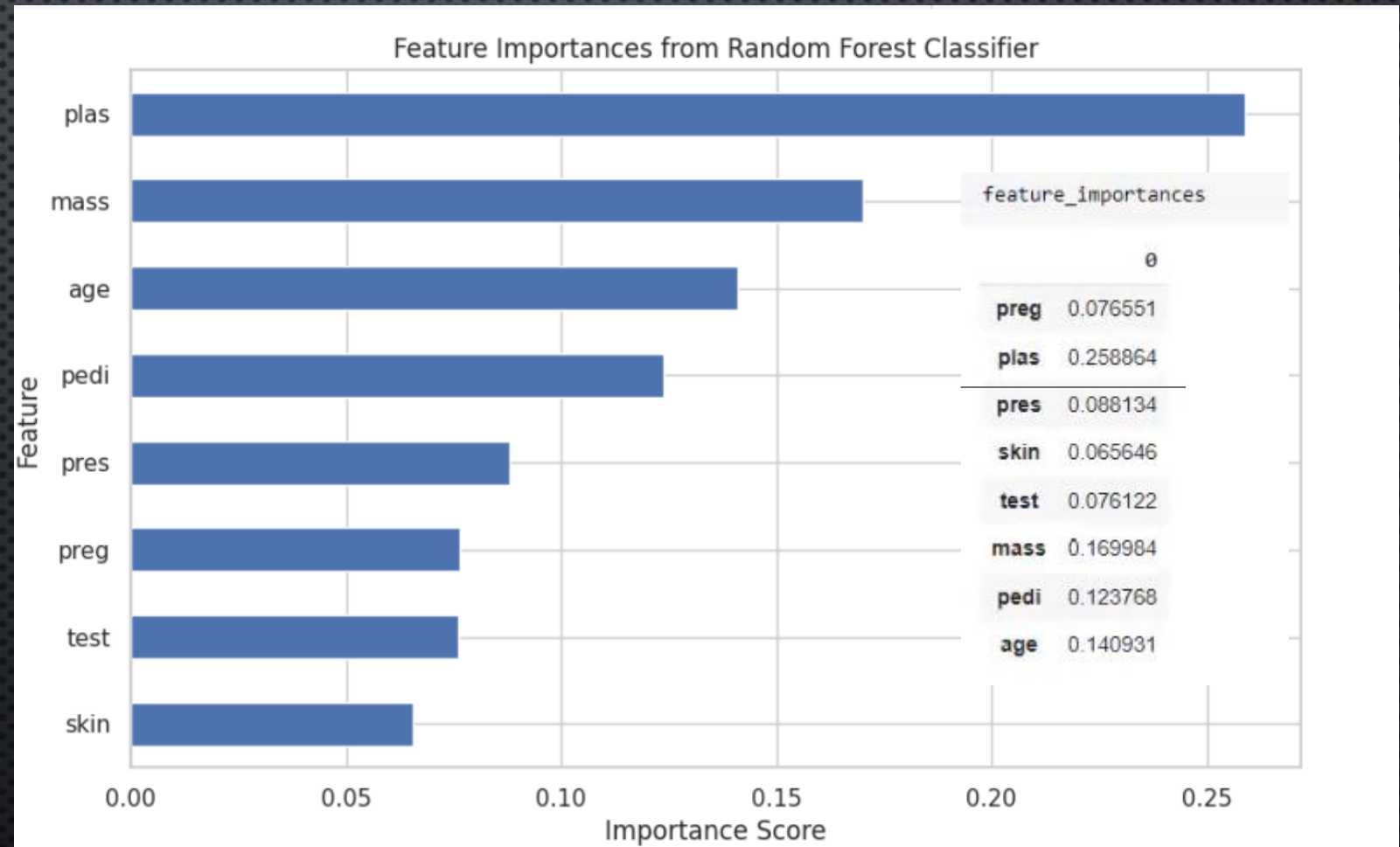
Regresión lineal

preg	0.064544
plas	0.034092
pres	-0.013873
skin	0.003283
test	-0.001803
mass	0.102617
pedi	0.627304
age	0.037062



Regresión lineal

preg	0.064544
plas	0.034092
pres	-0.013873
skin	0.003283
test	-0.001803
mass	0.102617
pedi	0.627304
age	0.037062



Cámbiele el tamaño al bosque (100 por D a 1000 ...que paso

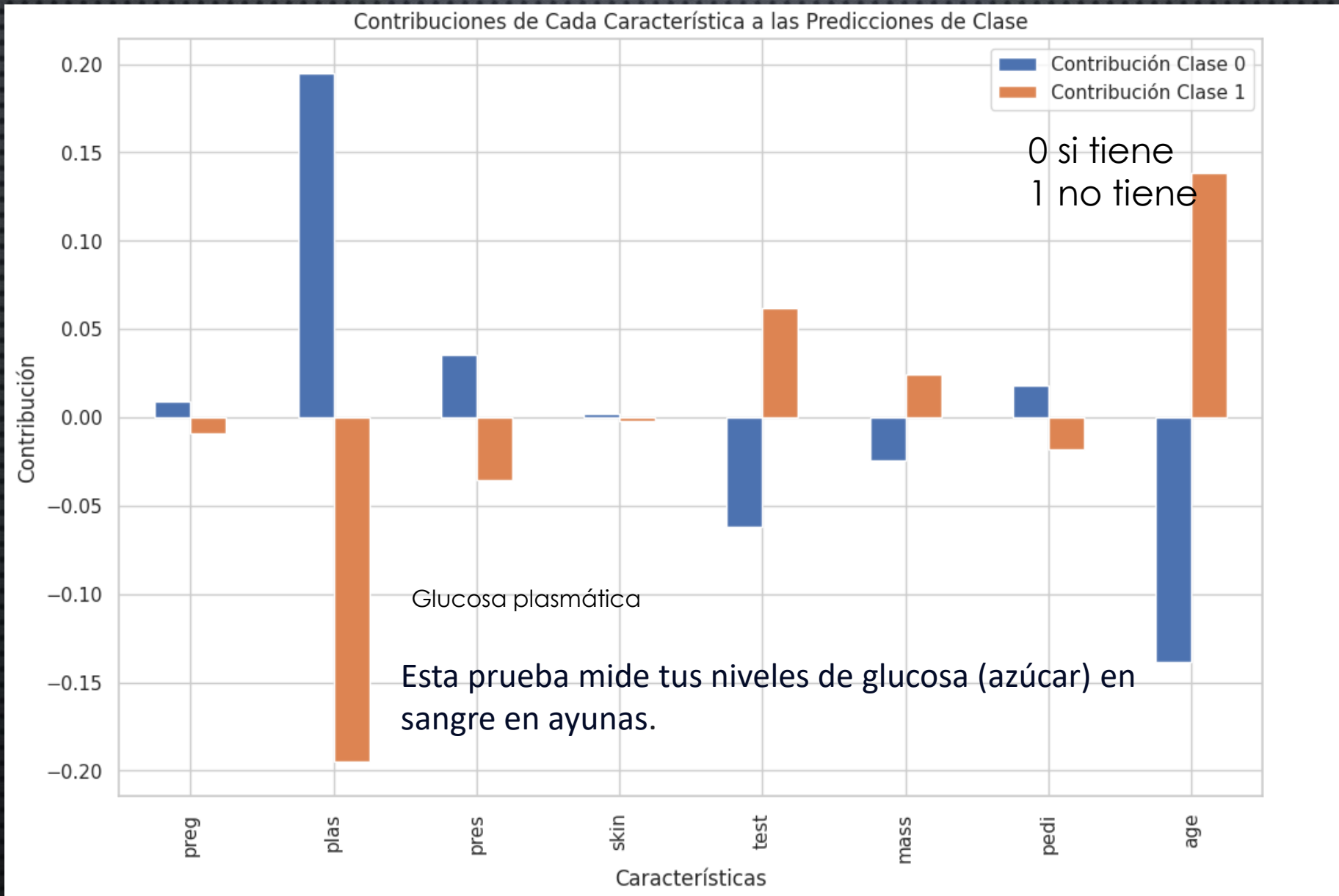

```
# Crear un DataFrame para visualizar mejor las contribuciones para ambas clases
contrib_df = pd.DataFrame(contributions[0], columns=['Contribución Clase 0', 'Contribución Clase 1'])
contrib_df.index = names[:8] # Nombres de las características
contrib_df = contrib_df.round(4) # Redondear para mejor visualización
```

Si accedo a contributions se cuanto contribuye cada variable a cada clase

Puedo observar pero de cada variable para cada clase

```
Predicción del modelo: [[0.69 0.31]]
Valor esperado (Bias): [[0.65462541 0.34537459]]
Contribuciones de cada característica a la predicción
```

	Contribución Clase 0	Contribución Clase 1
preg	0.0091	-0.0091
plas	0.1948	-0.1948
pres	0.0356	-0.0356
skin	0.0019	-0.0019
test	-0.0618	0.0618
mass	-0.0245	0.0245
pedi	0.0184	-0.0184
age	-0.1381	0.1381



PATOLOGIA DE CANCER DE MAMA: Características relacionadas con la morfología celular

Observaciones clave:

1. Principales características contribuyentes:

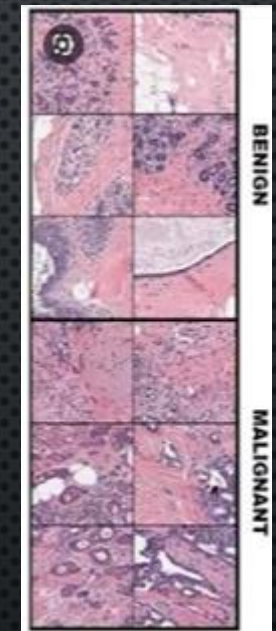
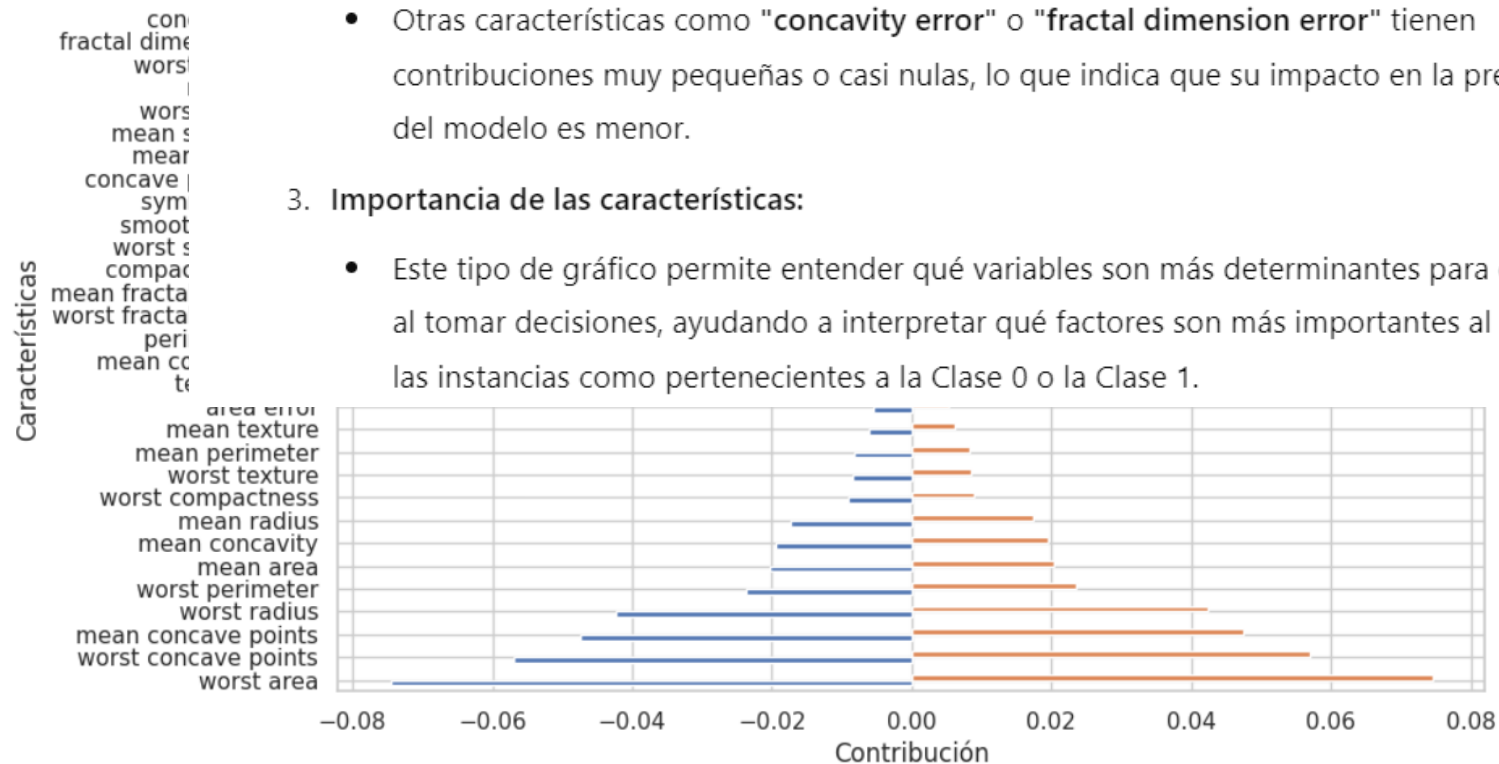
- Algunas características, como "**worst area**", "**worst concave points**", tienen una fuerte influencia tanto hacia la Clase 0 como hacia la Clase 1, dependiendo de su valor.

2. Características menos influyentes:

- Otras características como "**concavity error**" o "**fractal dimension error**" tienen contribuciones muy pequeñas o casi nulas, lo que indica que su impacto en la predicción del modelo es menor.

3. Importancia de las características:

- Este tipo de gráfico permite entender qué variables son más determinantes para el modelo al tomar decisiones, ayudando a interpretar qué factores son más importantes al clasificar las instancias como pertenecientes a la Clase 0 o la Clase 1.



Indican que estas características hacen que el modelo incline hacia la predicción de ser 0 o 1

PCA

```
# 4. PCA para la importancia de características
pca = PCA(n_components=X_train_california.shape[1]) # PCA con el mismo número
X_pca = pca.fit_transform(X_train_california)

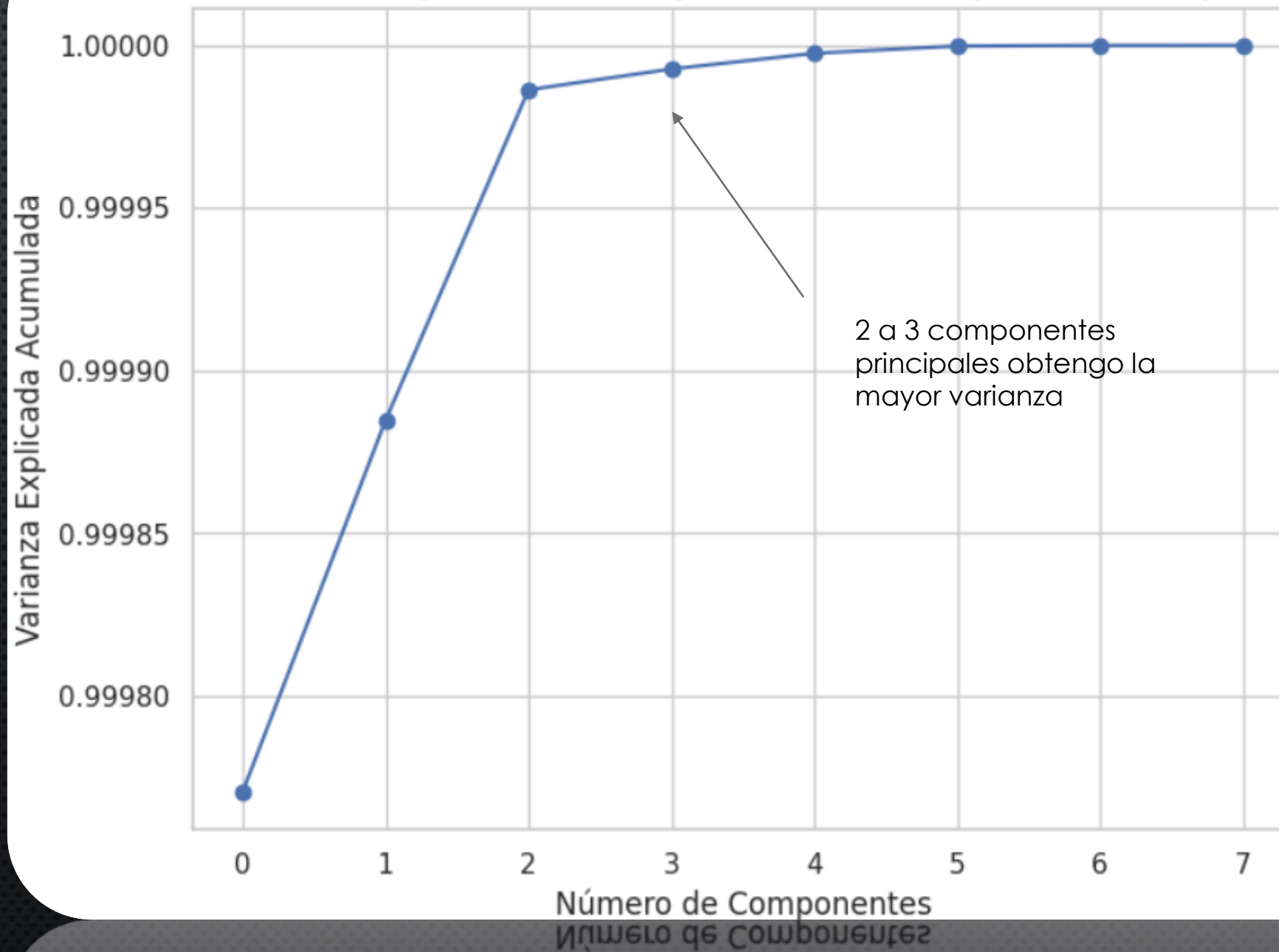
# Gráfico
plt.figure()
plt.plot(X_pca[:,0], X_pca[:,1])
plt.title('PCA plot of California housing data')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.grid(True)
plt.show()

# Importancia de las características
pca.co
```

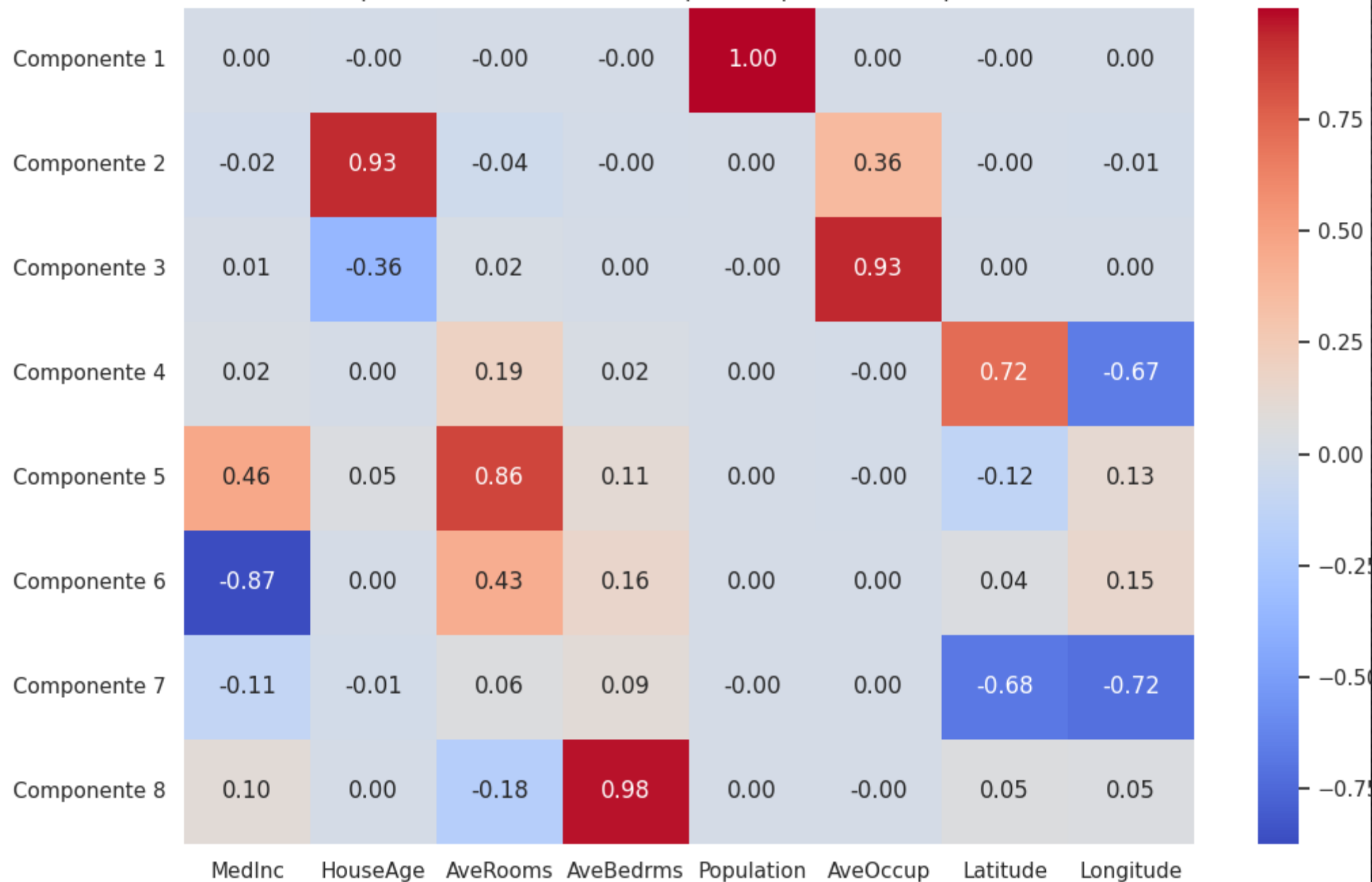
```
# 4. PCA para la importancia de características
pca = PCA(n_components=X_train_california.shape[1]) # PCA con el mismo número de componentes que características originales
X_pca = pca.fit_transform(X_train_california)
```

El Análisis de Componentes Principales (PCA) lo que hace es calcular una matriz de covarianza. A partir de esa matriz, se obtienen los autovectores y autovalores. Luego, se realiza una descomposición en valores singulares (SVD, por sus siglas en inglés) para identificar los componentes principales. El objetivo final es encontrar los ejes en los que se distribuye la máxima varianza de los datos.

Cumulative Explained Variance por Número de Componentes Principales



Importancia de Características por Componente Principal (PCA)



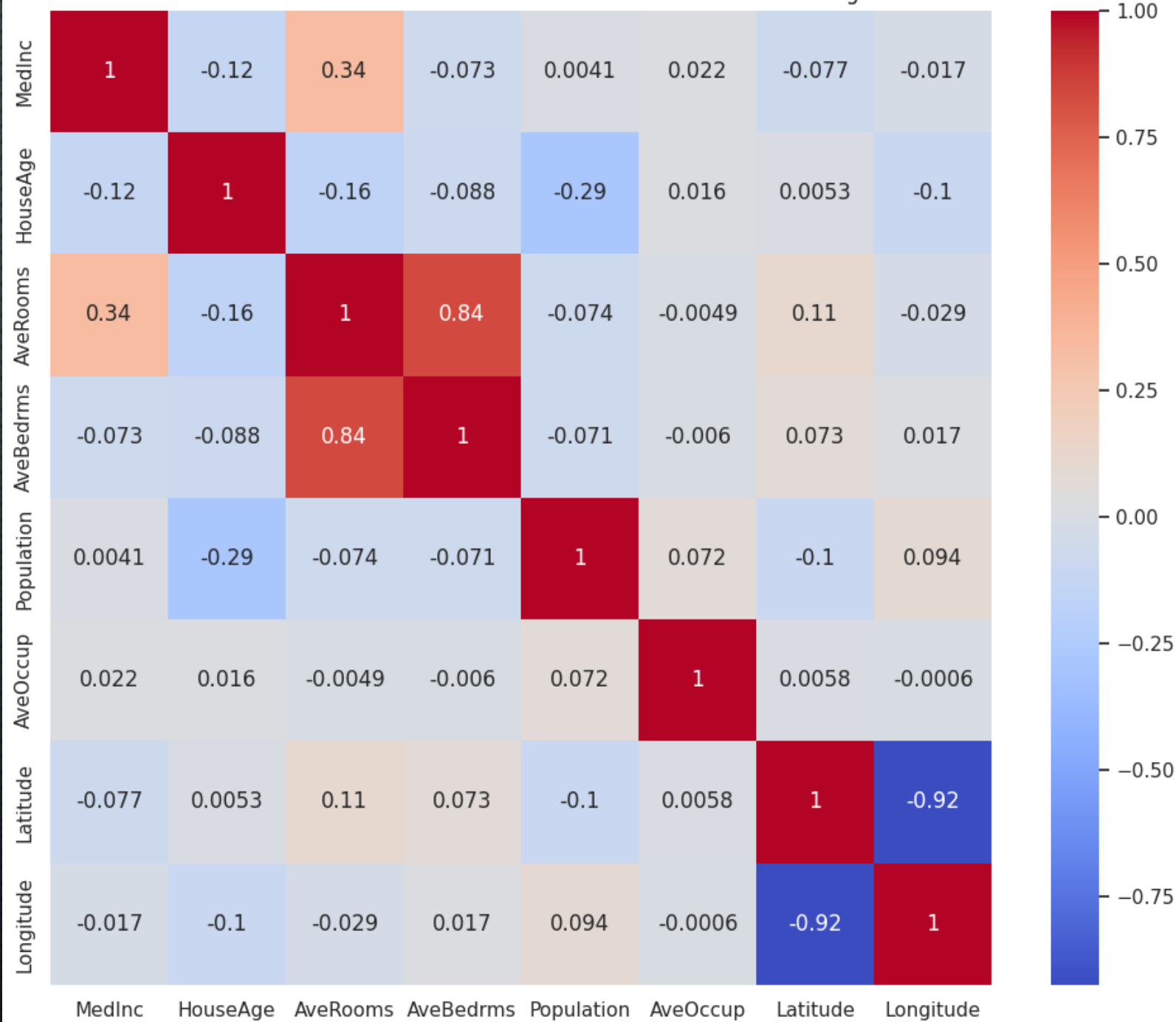
✓ 2. Técnicas de Feature selection

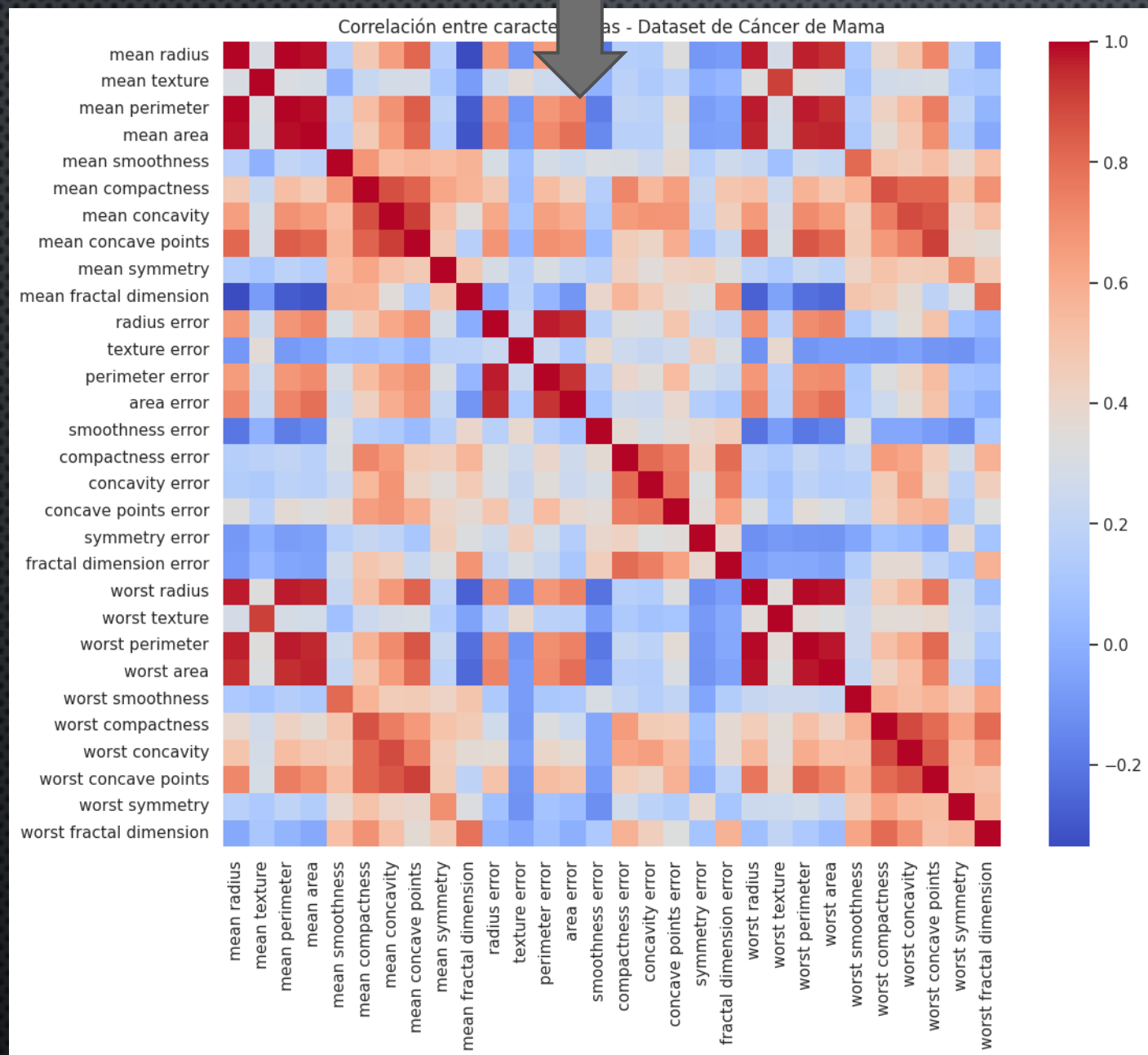
```
# 3. Evaluar la correlación entre características
plt.figure(figsize=(12, 10))
sns.heatmap(X_train_california.corr(), annot=True, cmap='coolwarm')
plt.title('Correlación entre características - Dataset de California Housing')
plt.show()

plt.figure(figsize=(12, 10))
sns.heatmap(X_train_cancer.corr(), annot=False, cmap='coolwarm')
plt.title('Correlación entre características - Dataset de Cáncer de Mama')
plt.show()
```

Dataset o X o df.corr() POR DEFECTO EVALUA LA CORRELACIÓN DE PEARSON ==> LINEAL

Correlación entre características - Dataset de California Housing

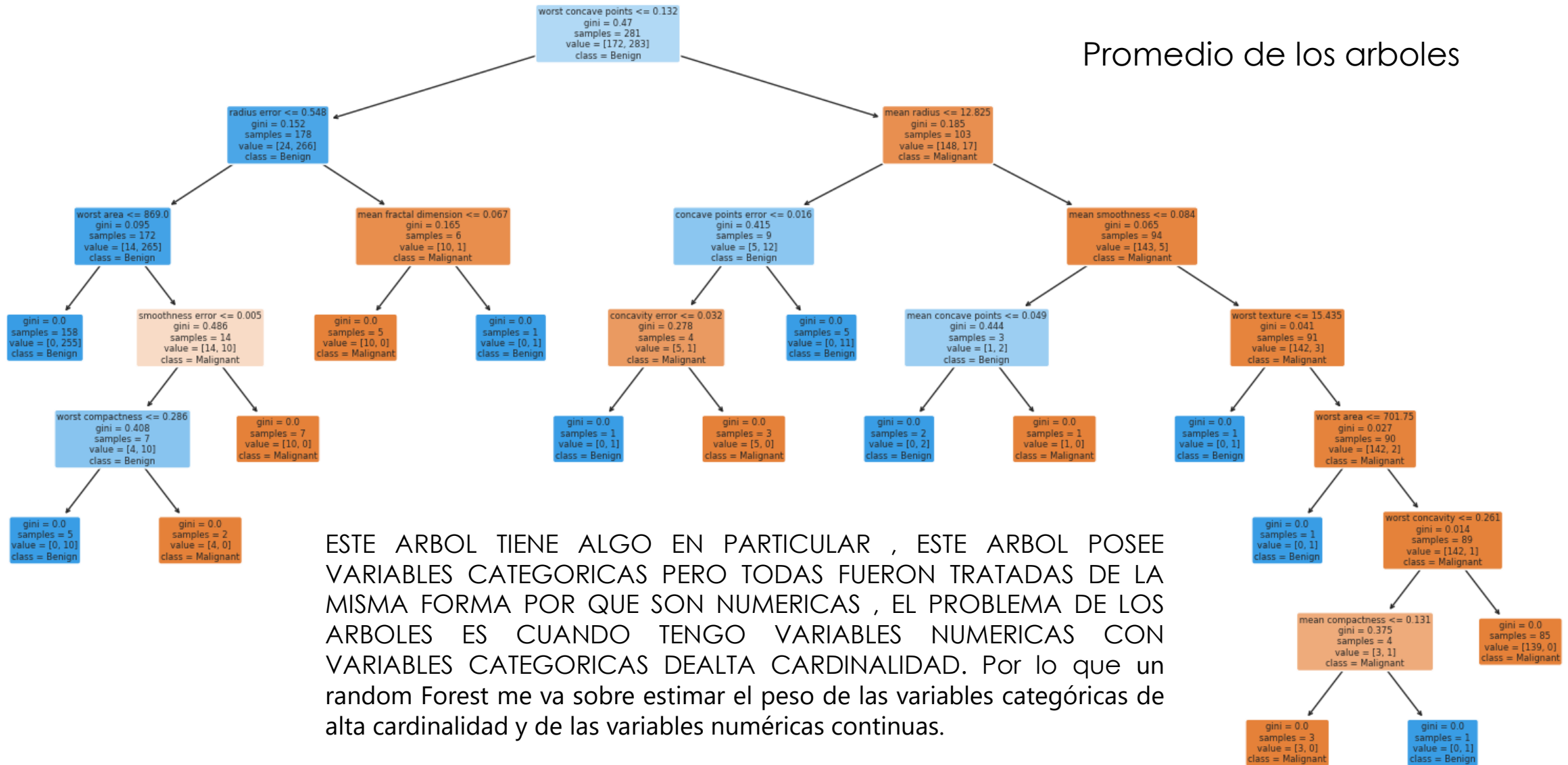




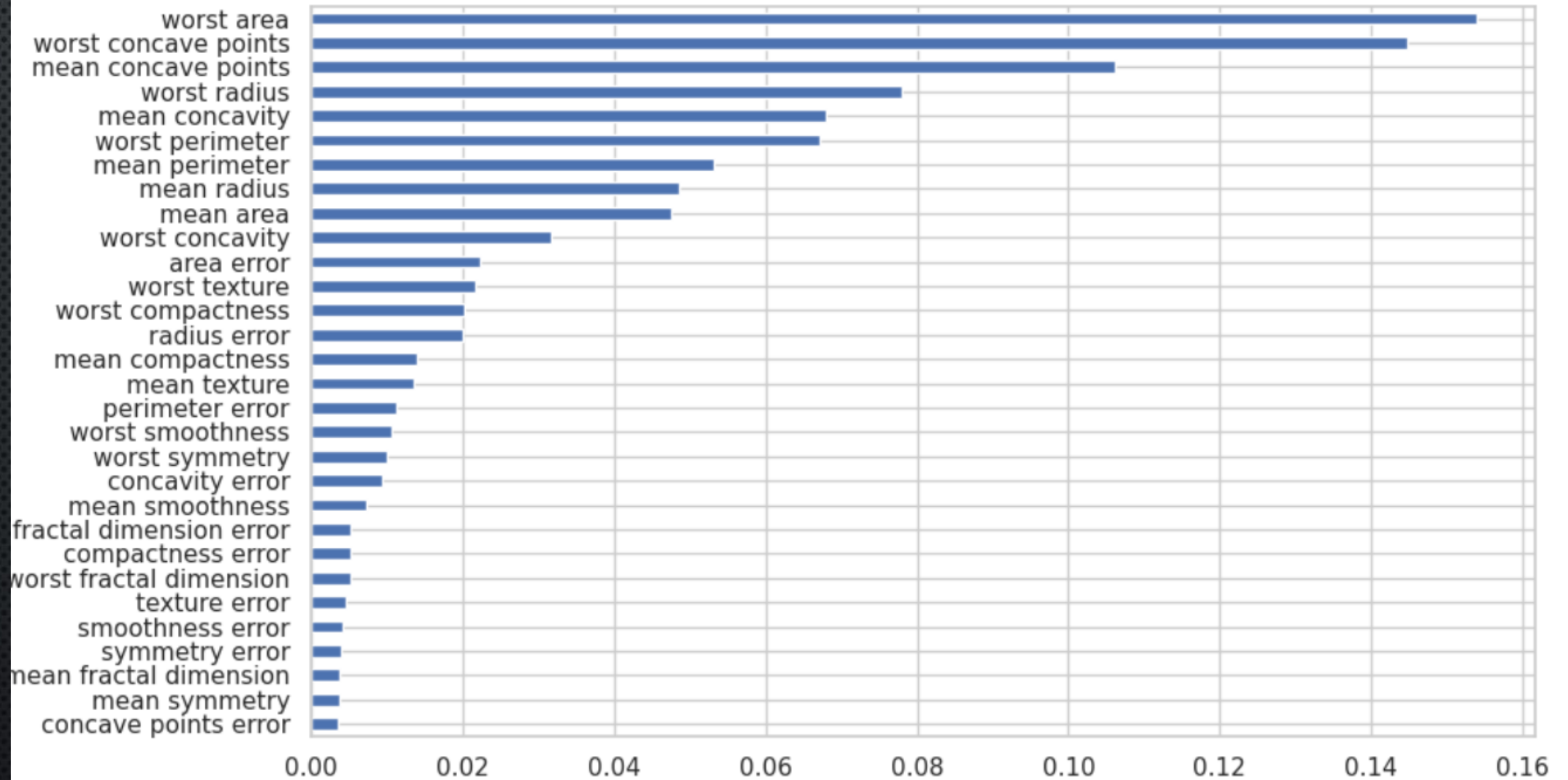
El radio máximo es un estimador utilizado por los patólogos y está claramente relacionado con otras variables como el área y el perímetro. En resumen, las asociaciones más fuertes que observamos son entre el radio, el perímetro y el área. Dado esto puedo decir que solo voy a usar el área o el radio de tumor

Árbol de Decisión en Random Forest

Promedio de los arboles



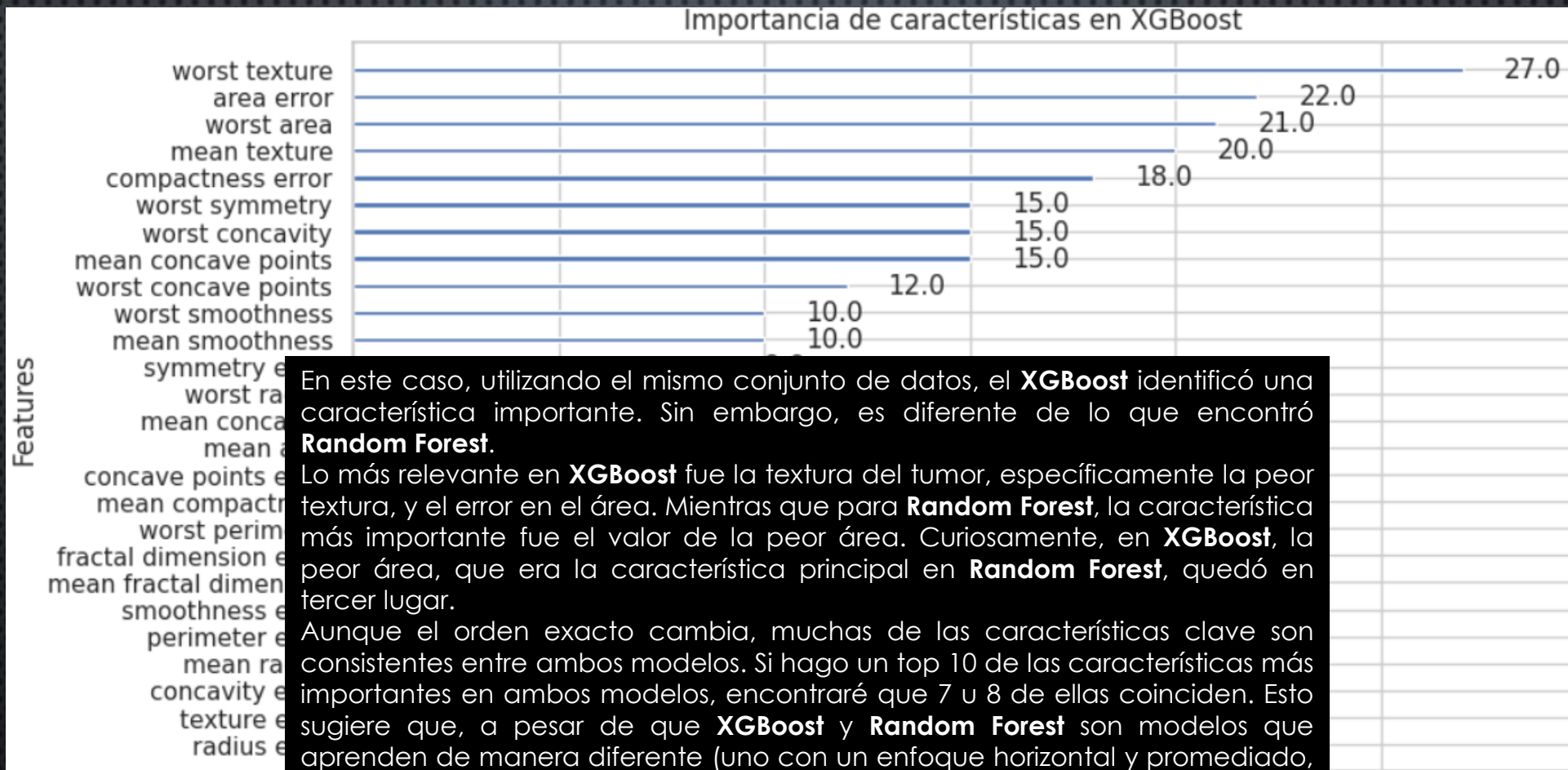
Importancia de características en Random Forest



```

from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from xgboost import XGBRegressor, XGBClassifier, plot_importance
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_classif, f_regression

```

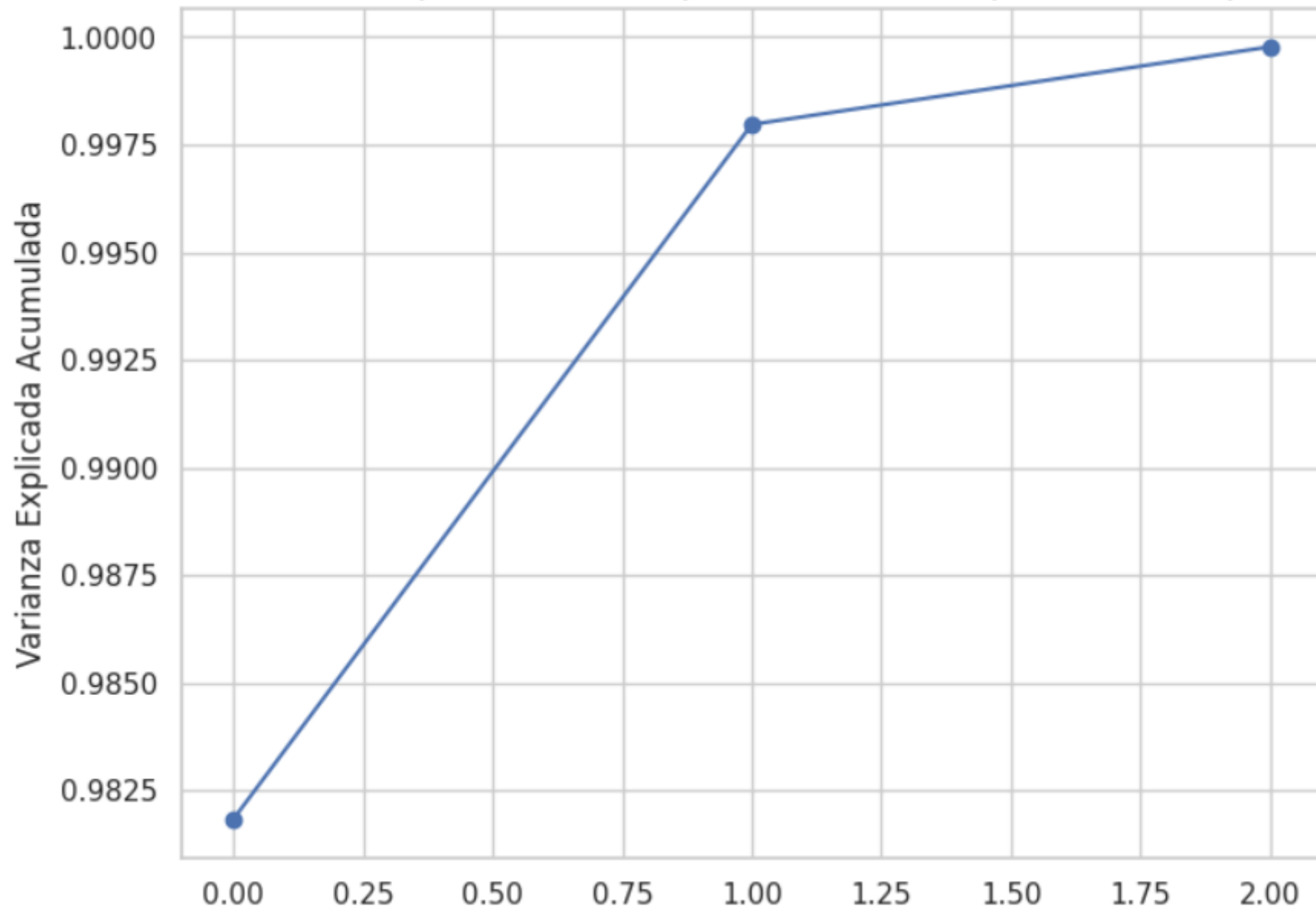


En este caso, utilizando el mismo conjunto de datos, el **XGBoost** identificó una característica importante. Sin embargo, es diferente de lo que encontró **Random Forest**.

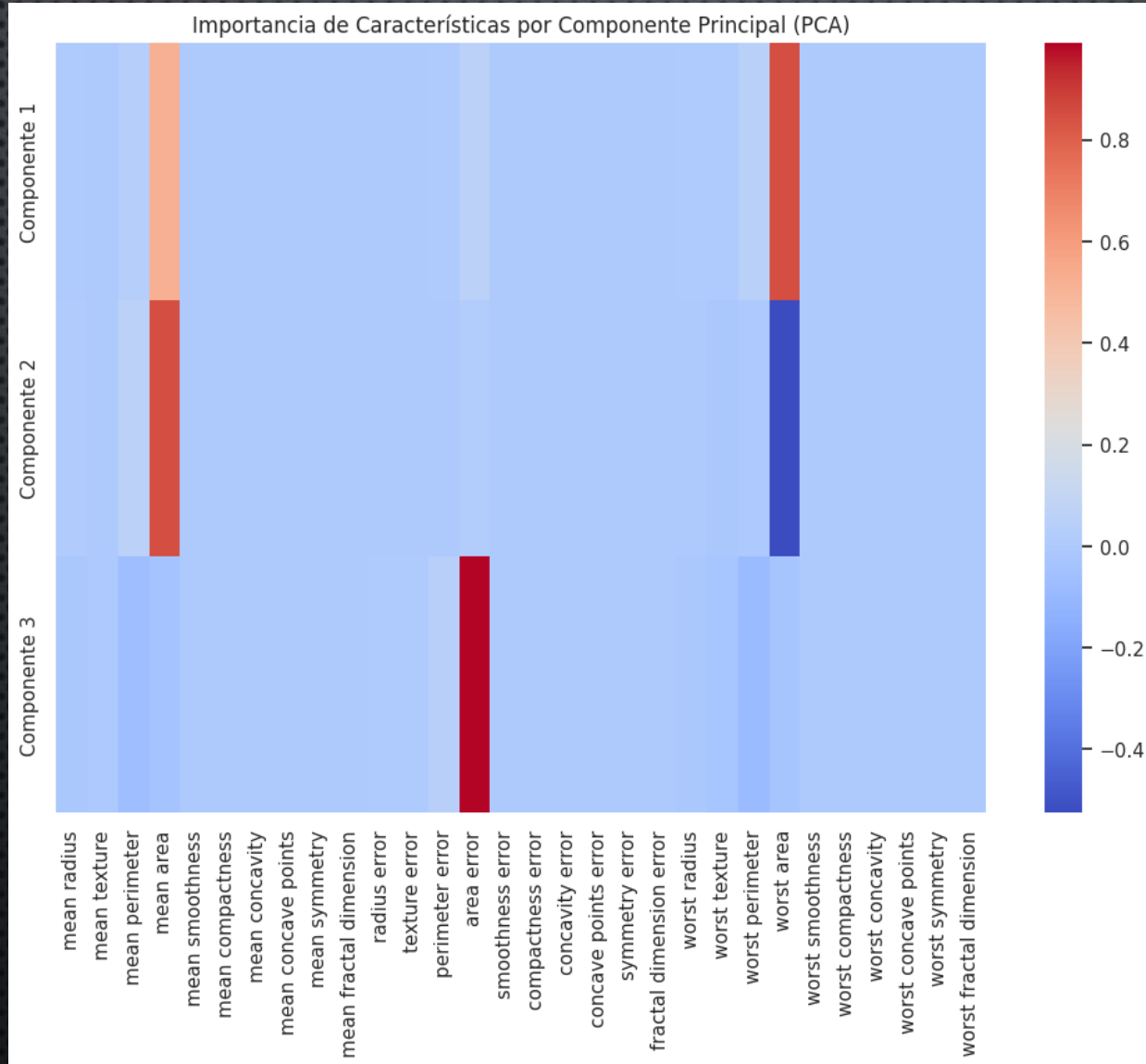
Lo más relevante en **XGBoost** fue la textura del tumor, específicamente la peor textura, y el error en el área. Mientras que para **Random Forest**, la característica más importante fue el valor de la peor área. Curiosamente, en **XGBoost**, la peor área, que era la característica principal en **Random Forest**, quedó en tercer lugar.

Aunque el orden exacto cambia, muchas de las características clave son consistentes entre ambos modelos. Si hago un top 10 de las características más importantes en ambos modelos, encontraré que 7 u 8 de ellas coinciden. Esto sugiere que, a pesar de que **XGBoost** y **Random Forest** son modelos que aprenden de manera diferente (uno con un enfoque horizontal y promediado, y el otro con un enfoque vertical y acumulativo), los resultados que producen

Cumulative Explained Variance por Número de Componentes Principales

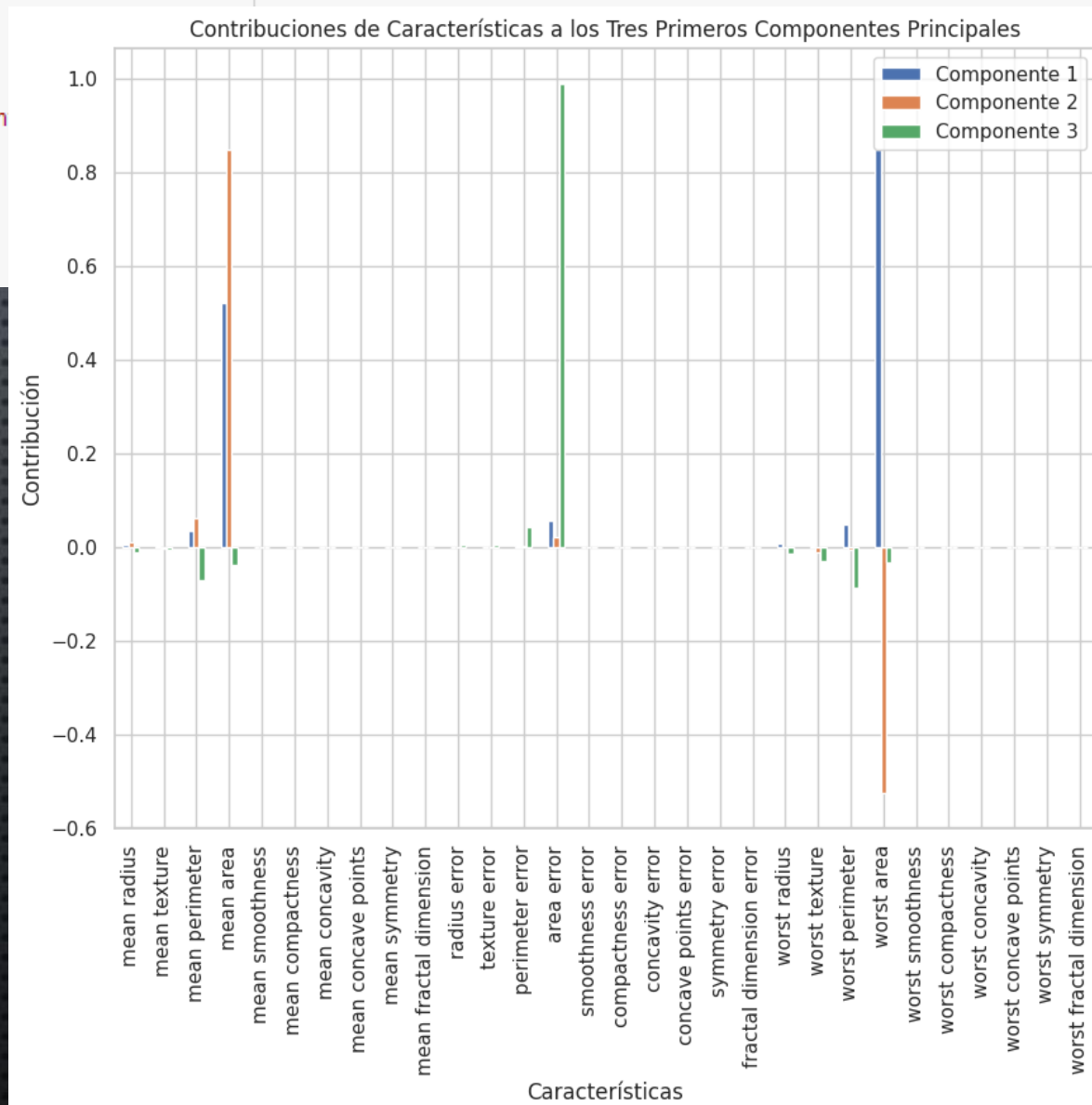



```
# 8. PCA para la importancia de características con 3 componentes principales al dataset para saber en que ejes tengo mayor varianza
pca = PCA(n_components=3) # Limitar a 3 componentes principales #USANDO 3 COMPONENTES PRINCIPALES.
X_pca = pca.fit_transform(X_train_cancer)
```



Gráfica de Varianza Explicada

```
plt.figure(figsize=(8, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o')
plt.title('Cumulative Explained Variance por Número de Componentes')
plt.xlabel('Número de Componentes')
plt.ylabel('Varianza Explicada Acumulada')
plt.grid(True)
plt.show()
```



FEATURE SELECTION

Utilizando **Random Forest**, **PCA** y **XGBoost**, he llegado a los mismos resultados en cuanto a la importancia de ciertas características. Esta coincidencia entre diferentes modelos me impulsa a seguir revisando varios enfoques y estrategias, ya que eso le otorga más robustez al análisis.

Este proceso se conoce como **selección de características**. Al aplicar diferentes métodos y modelos se puede identificar una tendencia consistente que ayuda a seleccionar las características más relevantes para el problema en cuestión. Esto refuerza la fiabilidad de los resultados y mejora la precisión del modelo.



Bosque aleatorio con búsqueda en cuadrícula

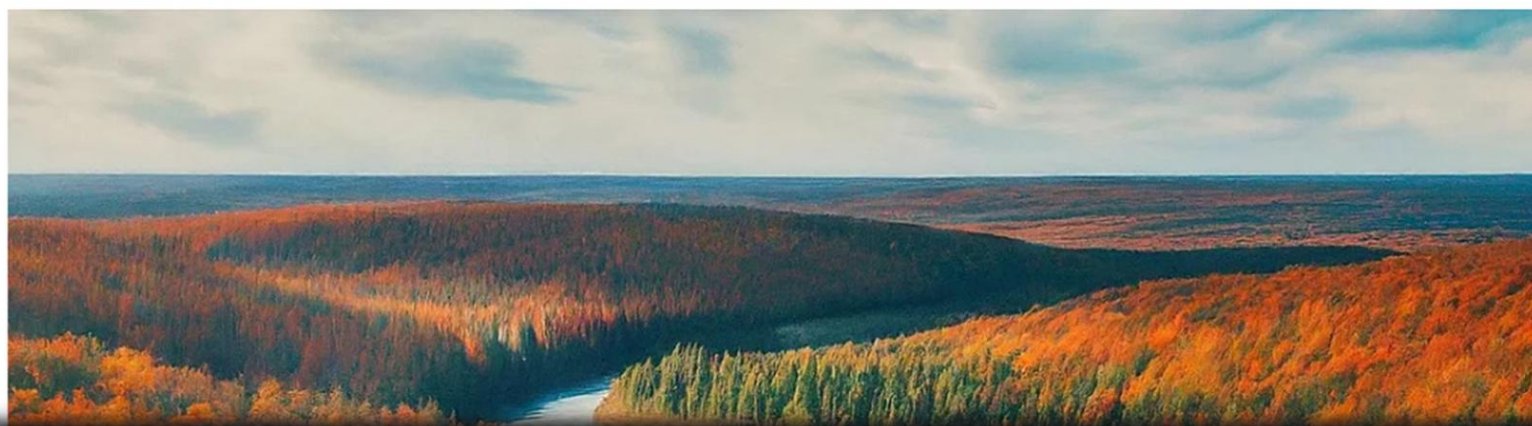


Seong Soon Mo · [Seguir](#)

Publicado en Villanos de la nube · 6 minutos de lectura · 13 de marzo de 2024



50



OTRAS LIBRERIAS :

✓ CATBOOST. Acepta variables categóricas

Es acepta variables categóricas, el RF Y XGB tiene que procesar la variable categórica , Codificarla en Binario entre 0 y 1

XGBoost. No acepta variables categóricas. MUY POTENTE. REQUIERE FINE TUNING.(Boosting, no promedia, vertical)

También toca codificarla (Getdummy- promedio)

Random forest. No acepta variables categóricas. No requiere fine tuning (ensemble promedio baggin)

one hot endoding. get dummies ==> Si tengo variables categoricas (No binarias)

