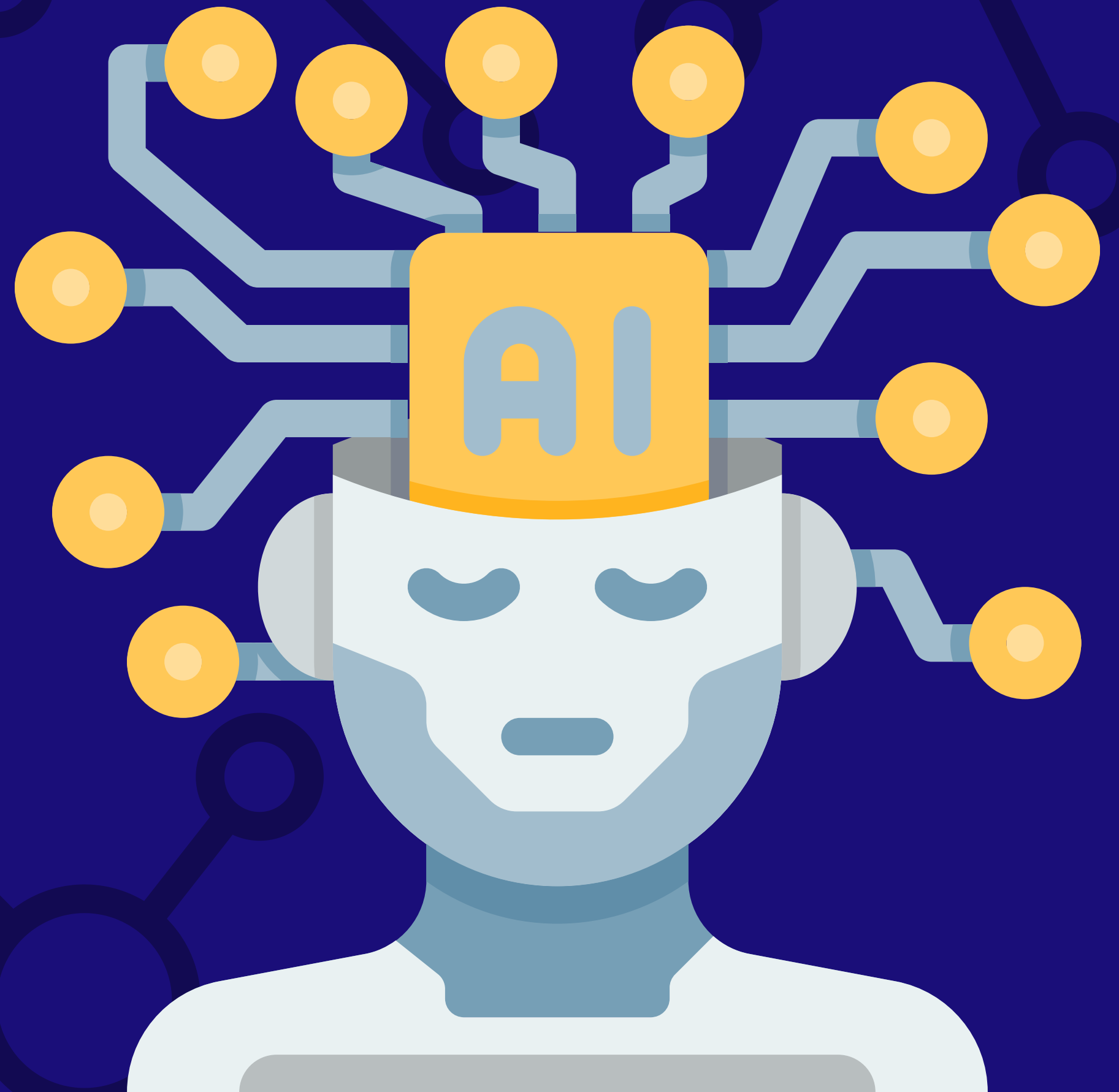
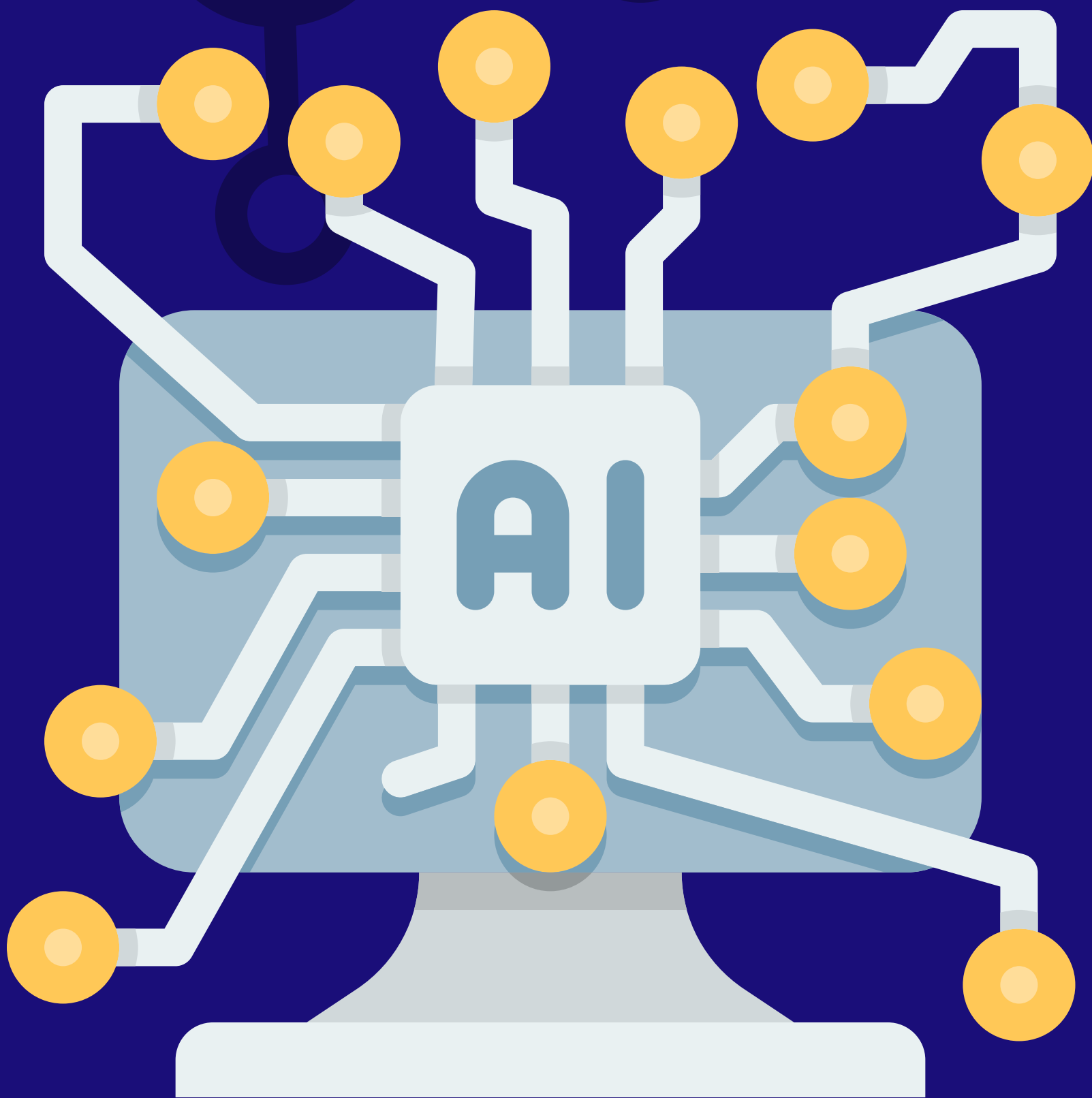


# Modelo de clasificación de homínidos

Teresa Terol Díez.  
Bootcamp Data Science. THE BRIDGE






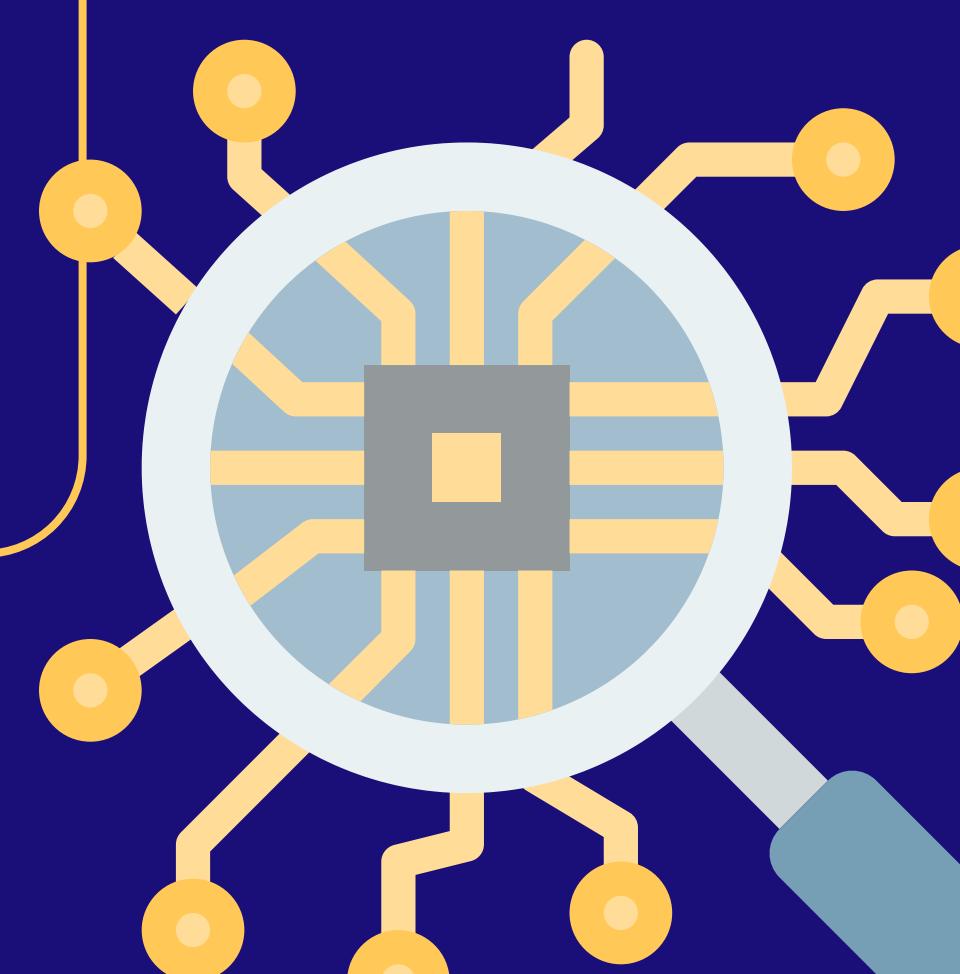
## Índice

- 01.** Introducción
- 02.** EDA
- 03.** Feature engineering
- 04.** Modelos
- 05.** Modelo final
- 06.** Evaluación del modelo
- 07.** Conclusiones



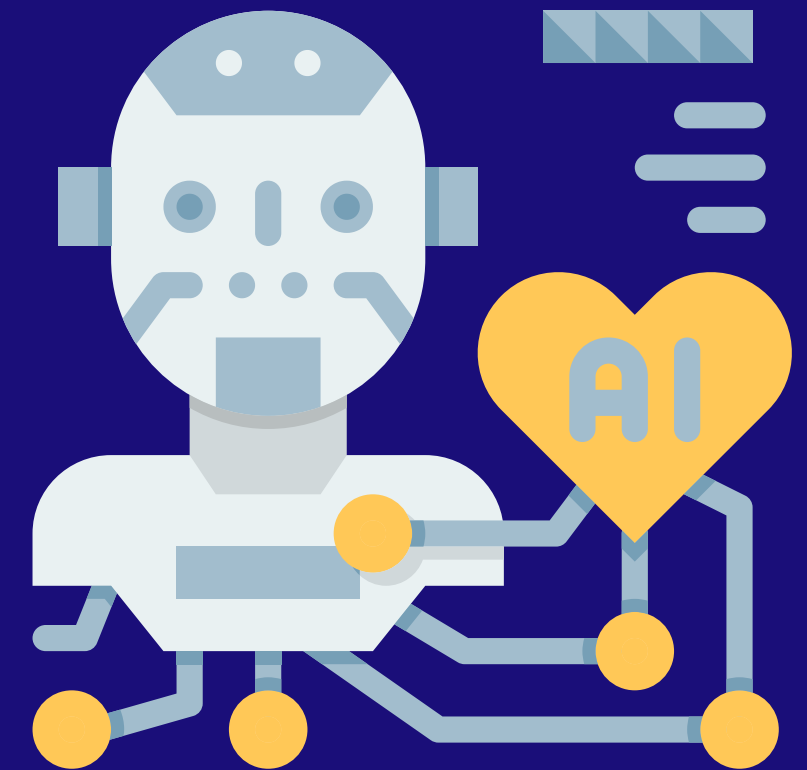
## 1.INTRODUCCIÓN

A través de la paleontología, los investigadores pueden reconstruir la historia evolutiva de diferentes especies y comprender mejor cómo ha cambiado nuestro planeta a lo largo del tiempo. Sin embargo, esta ciencia no está exenta de desafíos, especialmente cuando se trata de analizar grandes cantidades de datos y tomar decisiones basadas en la información obtenida. Es aquí donde la inteligencia artificial (IA) juega un papel fundamental, ya que puede ayudar a los paleontólogos a acelerar sus investigaciones y descubrir nuevos patrones de manera más eficiente.



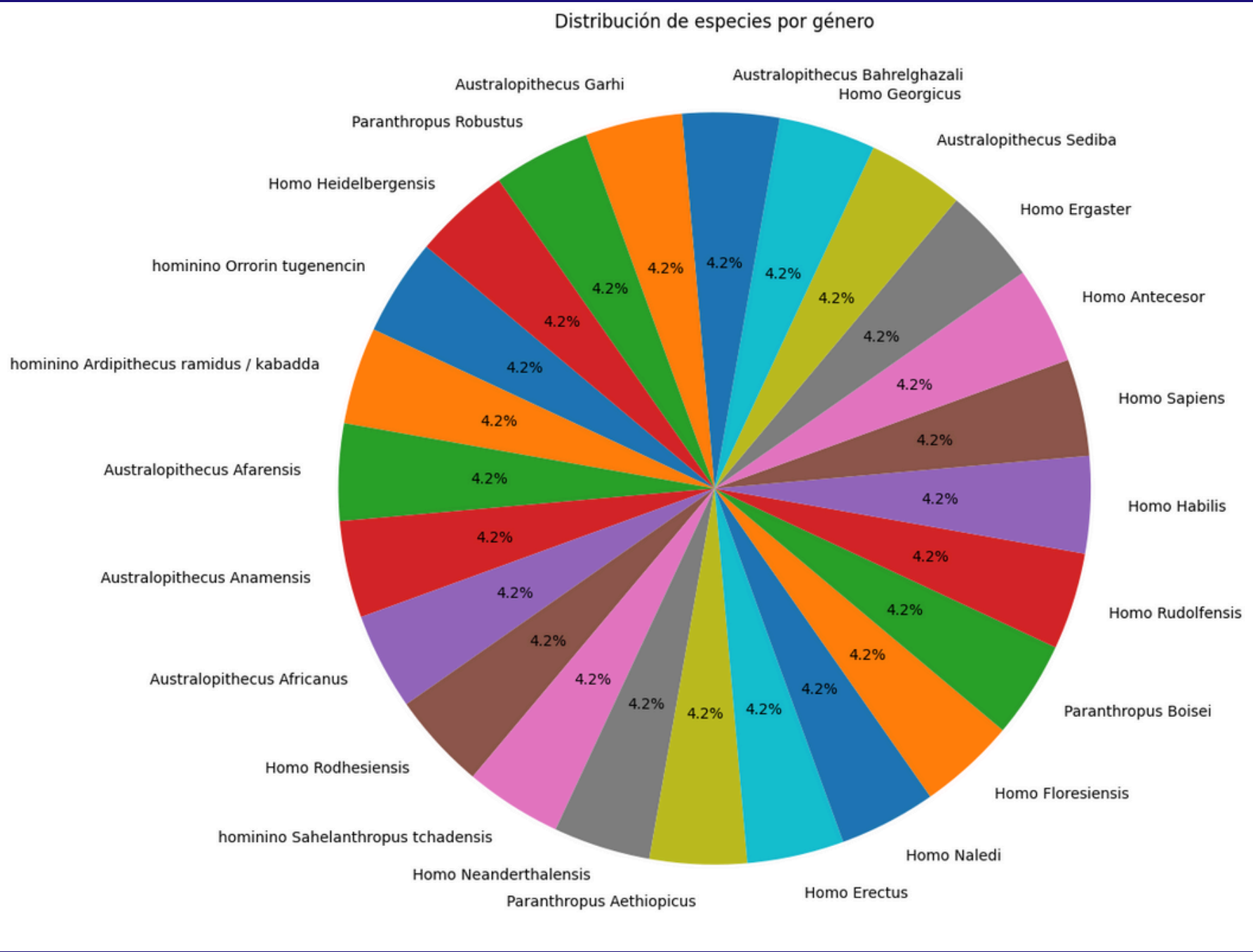
Para abordar esta problemática he desplegado un modelo de clasificación donde podemos predecir el género y especie de los diferentes homínidos en función de diferentes variables que se recogen cada vez que se encuentran nuevos fósiles, como por ejemplo:

- **Localización de los restos** (zona, hábitat)
- **Dieta**
- **Migración**
- **Tecnología y tipo de herramientas**
- **Características anatómicas:** Altura, capacidad craneal, grado de bipedismo, forma de la mandíbula, tamaño y forma de incisivos y caninos, prognatismo, grado de protuberancia del Torus Supraorbital, verticalidad frontal, posición del Foramen Mágnum, grosor del esmalte dental, grado de dimorfismo sexual, forma de la cadera, o forma de los pies y brazos.

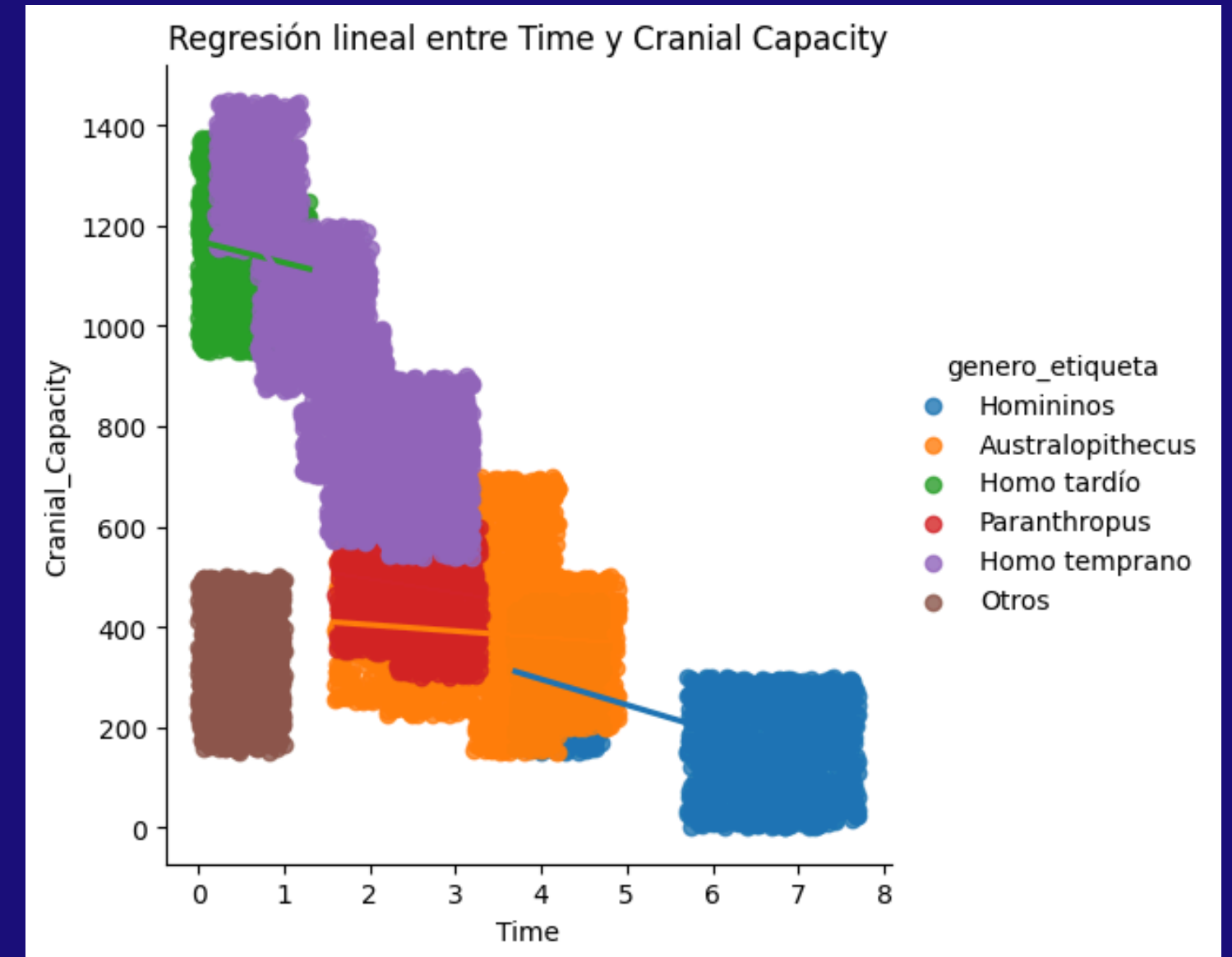
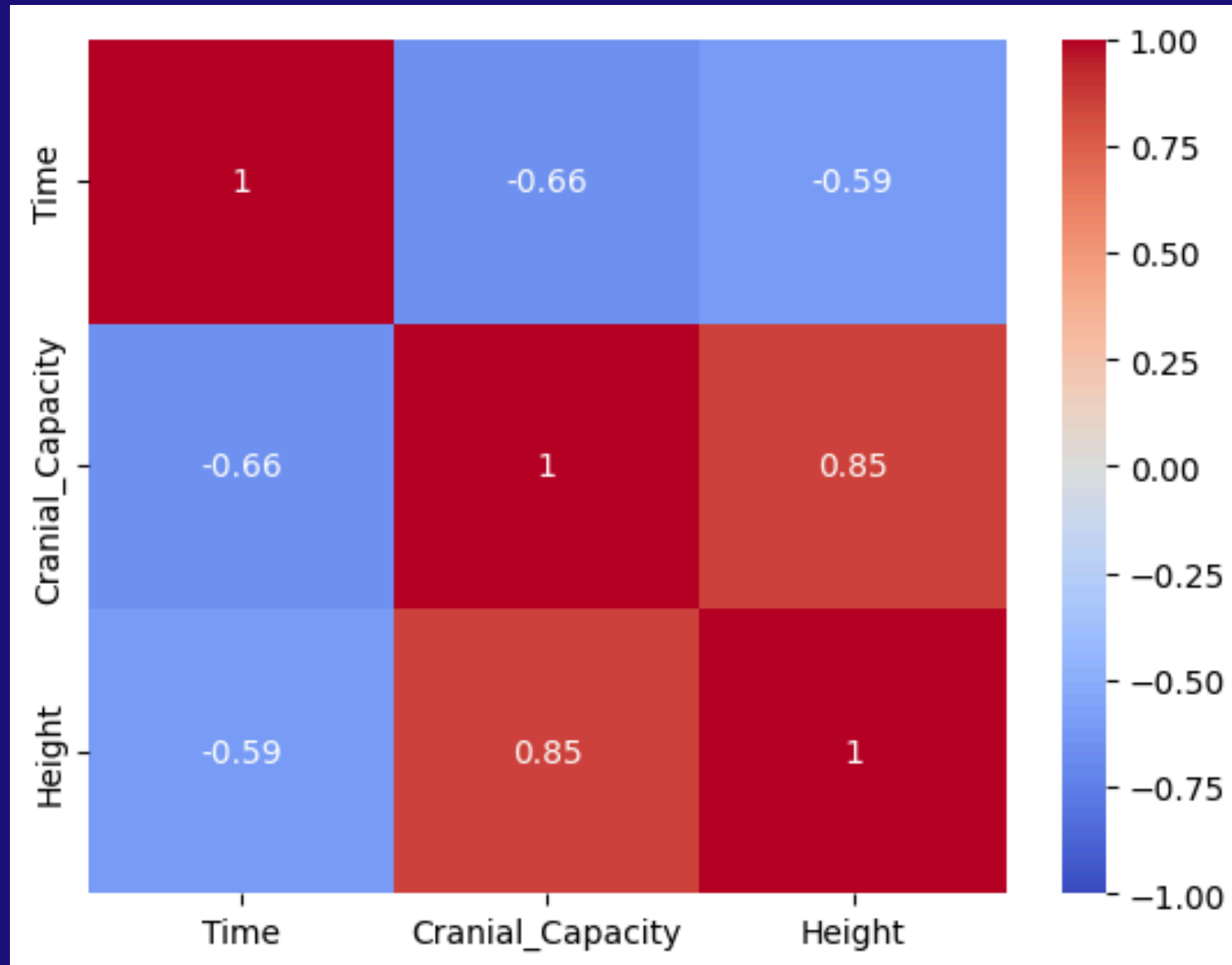


## 02. EDA:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12000 entries, 0 to 11999
Data columns (total 28 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Genus_&_Specie                        12000 non-null  object
1   Time                                  12000 non-null  float64
2   Location                              12000 non-null  object
3   Zone                                  12000 non-null  object
4   Current_Country                       12000 non-null  object
5   Habitat                               12000 non-null  object
6   Cranial_Capacity                      12000 non-null  float64
7   Height                                12000 non-null  float64
8   Incisor_Size                          12000 non-null  object
9   Jaw_Shape                             12000 non-null  object
10  Torus_Supraorbital                    12000 non-null  object
11  Prognathism                           12000 non-null  object
12  Foramen_Mágnum_Position               12000 non-null  object
13  Canine Size                           12000 non-null  object
14  Canines_Shape                         12000 non-null  object
15  Tooth_Enamel                          12000 non-null  object
16  Tecno                                 12000 non-null  object
17  Tecno_type                            12000 non-null  object
18  biped                                 12000 non-null  object
19  Arms                                  12000 non-null  object
...
26  Migrated                              12000 non-null  object
27  Skeleton                              12000 non-null  object
dtypes: float64(3), object(25)
```



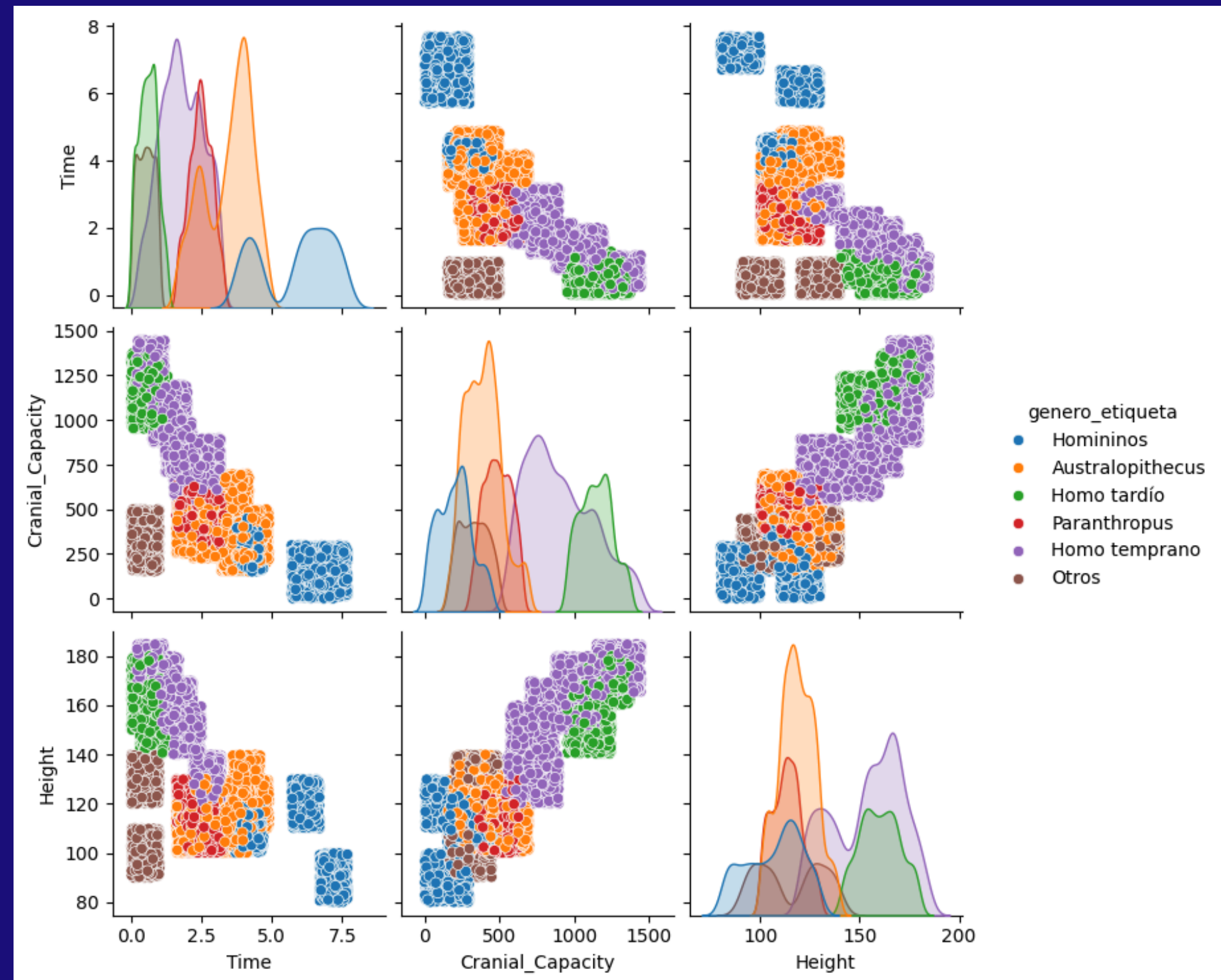
# CORRELACIÓN ENTRE LAS VARIABLES SIN FEATURE ENGINEERING

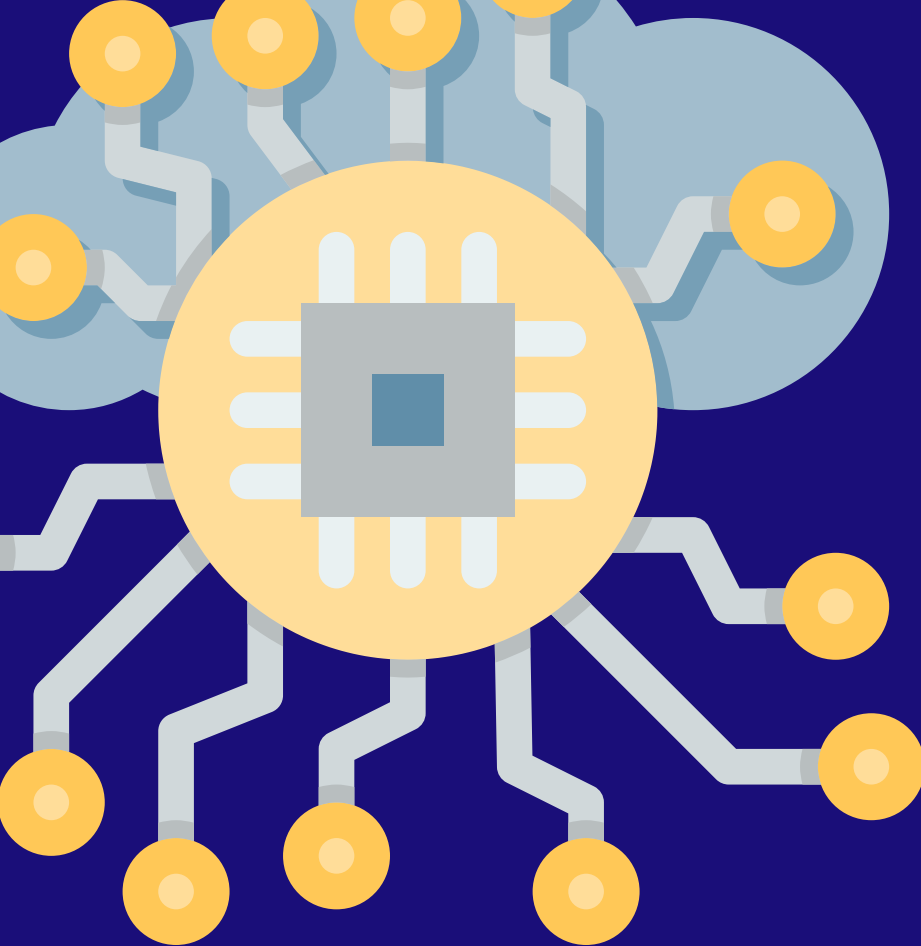




## Gráfico de correlación y distribución de nuestras variables numéricas

Podemos observar cómo a mayor antigüedad menor capacidad craneal y menor altura





### 3. FEATURE ENGINEERING

Para realizar el feature engineering óptimo, he transformado a numéricas todas las variables categóricas teniendo en cuenta las características evolutivas más antiguas y y dándoles un valor mas próximo a 0 y al contrario con las características evolutivas más modernas.

He optado por usar la técnica de Mapping para las variables categóricas ordinales que quería asignar en función de su antigüedad y LableEncoder para las que eran categóricas cardinales.



# Lable Encoder

```
label_encoder = LabelEncoder()  
df2['Location_encoded'] = label_encoder.fit_transform(df2['Location'])
```

```
print("Valores únicos en 'Location':", df2['Location'].unique())  
print("Valores codificados numéricamente:", df2['Location_encoded'].unique())
```

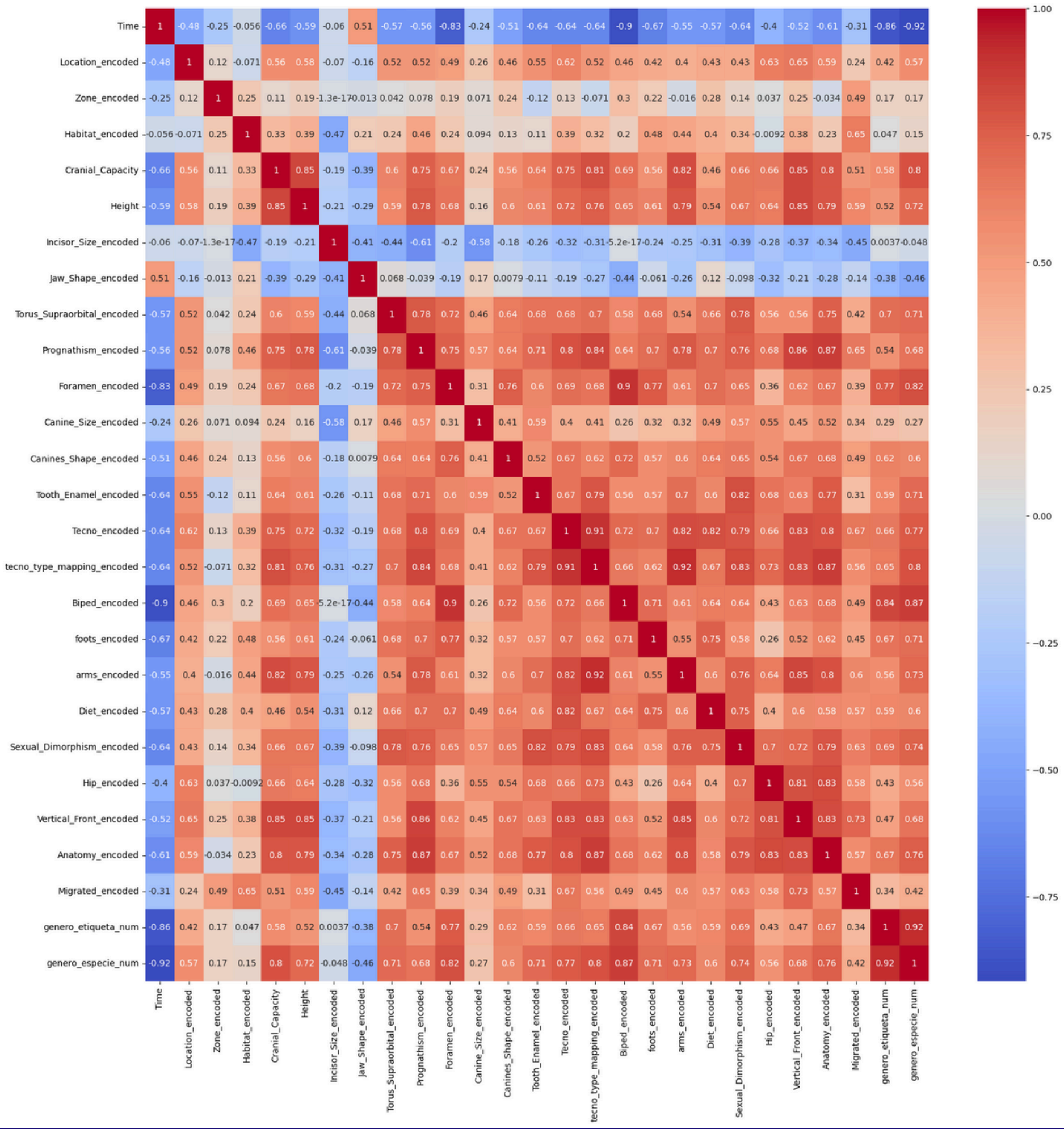
```
Valores únicos en 'Location': ['Africa' 'Europa' 'Asia ']  
Valores codificados numéricamente: [0 2 1]
```

# Mapping

```
prognathism_mapping = {  
    'absent': 5,  
    'reduced': 4,  
    'medium': 3,  
    'medium-high': 2,  
    'high': 1,  
    'very high': 0  
}
```

```
df2['Prognathism_encoded'] = df2['Prognathism'].map(prognathism_mapping)
```

Tras convertir todas las variables a valores numéricas, vemos que la matriz de correlación de nuestras variables es mucho más compleja y se puede observar una alta multicolinealidad entre nuestras variables, sin haberlas relacionado con nuestro target.



## 4.MODELOS

### Random Forest Classifier

- Es altamente interpretable
- Menos propenso al sobreajuste en comparación con un solo árbol de decisión.
- Capaz de manejar automáticamente las características más importantes para la clasificación.

### K-Nearest Neighbors

- Es simple.
- No es tan influenciado por valores atípicos
- Tiende al overfitting cuando hay muchas características.
- Sensibilidad a la dimensionalidad.

### Linear Regressor Classifier

- Es simple y eficaz, trabaja como un modelo de función sigmoide.
- Aun que es más usado como un modelo de clasificación binaria tiene parámetros para trabajar con problemas multiclase (One vs. Rest, One vs. One).

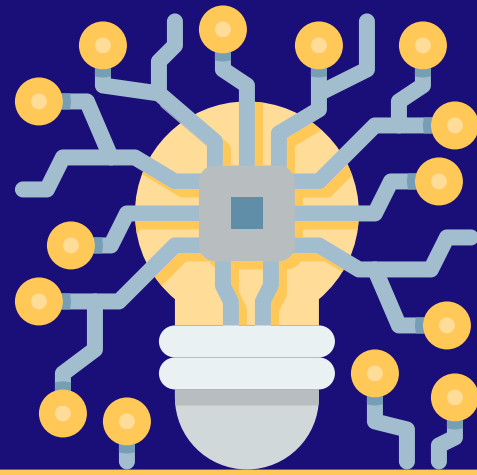
### Support Vector Machine

- Más costoso computacionalmente en comparación con algunos otros algoritmos de aprendizaje supervisado.
- Modelo robusto.
- Menos sensible a datos atípicos.

### Gradient Boosting Classifier

- Puede manejar automáticamente la importancia de las características.
- Modelo robusto.
- Puede ser más costoso computacionalmente y requerir más tiempo de entrenamiento .





## PREPROCESADO: PCA Y ESCALADO

He incluido como pre-procesado en todos los pipelines de mis modelos el Análisis de los componentes principales (PCA) para ver si podía reducir la multicolinealidad y la dimensionalidad de mis variables.

También he incluido un paso para el escalado de los datos de manera que estén en las unidades estandarizadas para ayudar a los modelos a encontrar patrones.

RandomizeSearch para hiperparametrizar mi modelo de manera óptima.

```

steps = [
    ('scaler', StandardScaler()),
    ('pca', PCA()),
    ('classifier', RandomForestClassifier(random_state=42))
]

pipeline = Pipeline(steps)

param_dist = {
    'scaler': [None, StandardScaler(), MinMaxScaler()],
    'pca__n_components': [24,25],
    'classifier__n_estimators': [100, 500, 1000],
    'classifier__max_depth': [3,5],
    'classifier__max_leaf_nodes': [16,17,18]
}

random_search = RandomizedSearchCV(pipeline, param_distributions=param_dist, cv=5, n_iter=10,n_jobs=-1, random_state=42,verbose= 2)

rs=random_search.fit(X, y)

best_score = random_search.best_score_
best_params = random_search.best_params_

print("Best Score:", best_score)
print("Best Parameters:", best_params)

```

✓ 37.8s

Fitting 5 folds for each of 10 candidates, totalling 50 fits

Best Score: 1.0

Best Parameters: {'scaler': MinMaxScaler(), 'pca\_\_n\_components': 25, 'classifier\_\_n\_estimators': 100, 'classifier\_\_max\_leaf\_nodes': 17, 'classifier\_\_max\_depth': 5}

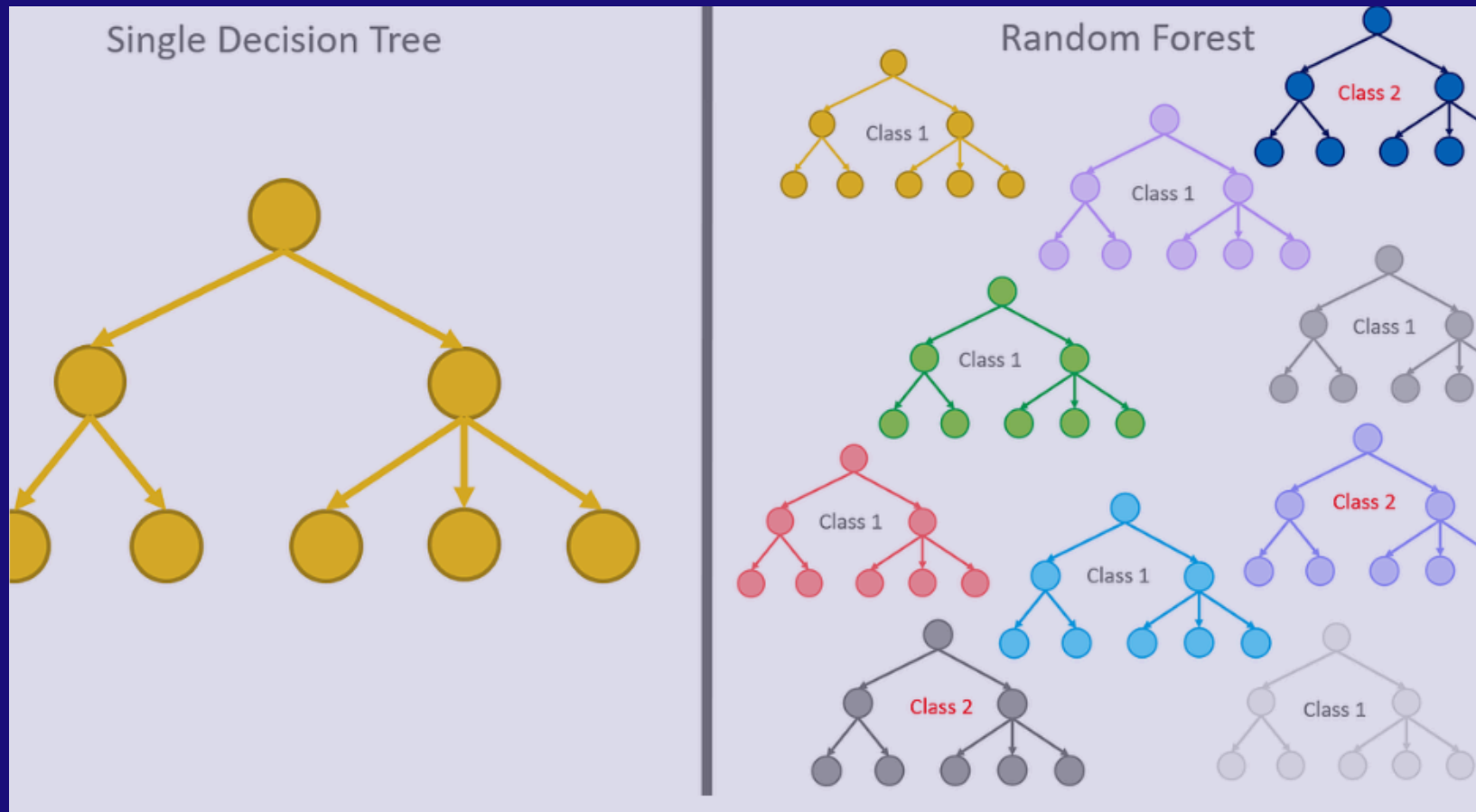
## 5. Modelo final: Random Forest Classifier

He optado por este modelo por su buena interpretabilidad, por su relativamente bajo costo computacional, y sus buenos resultados en las métricas de evaluación.

Todos los modelos han tenido unas métricas muy altas debido a que el problema que estamos abordando es altamente determinista y el conjunto de datos es muy limpio y está bien estructurado.

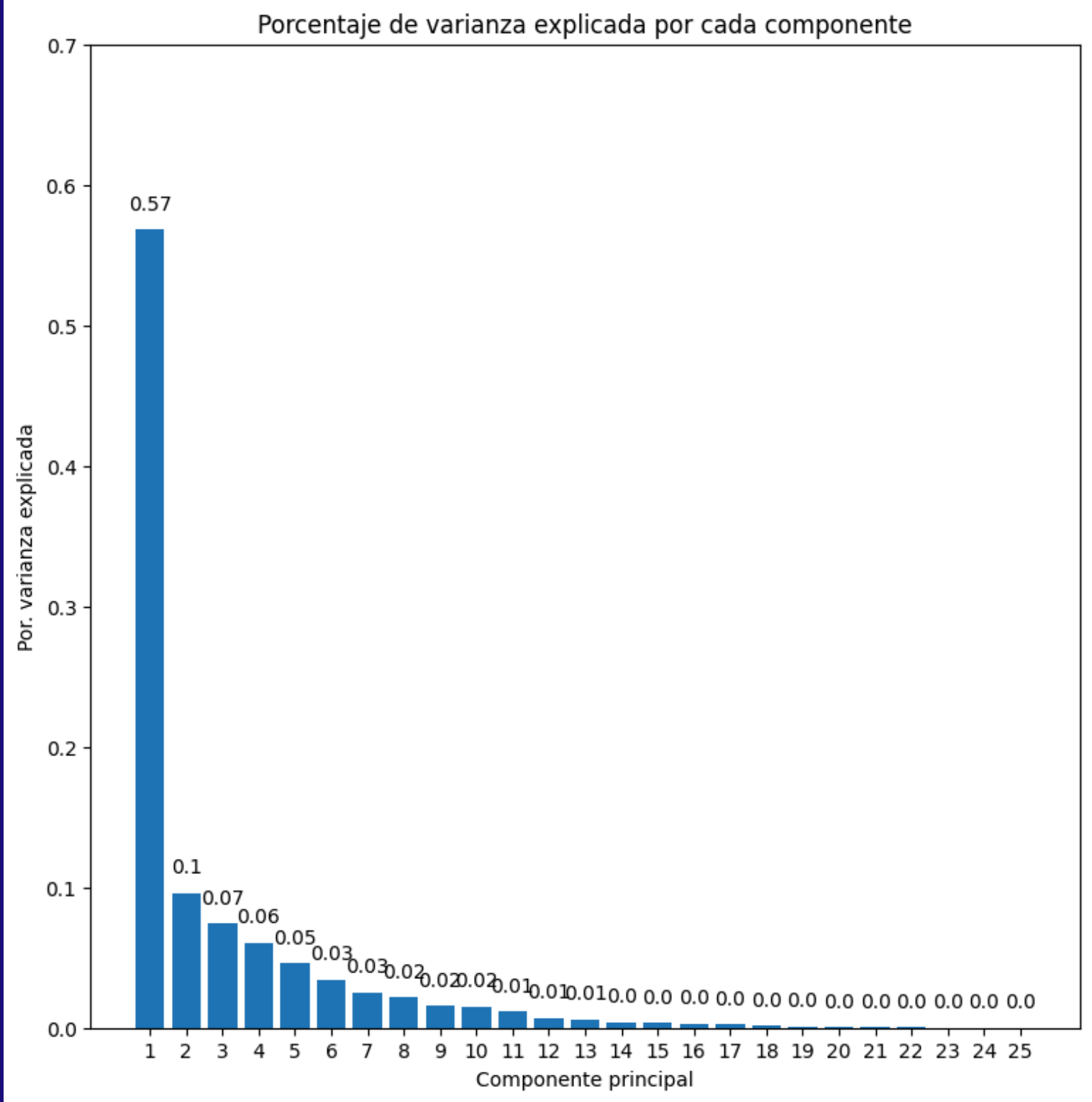


# Mejores parámetros:

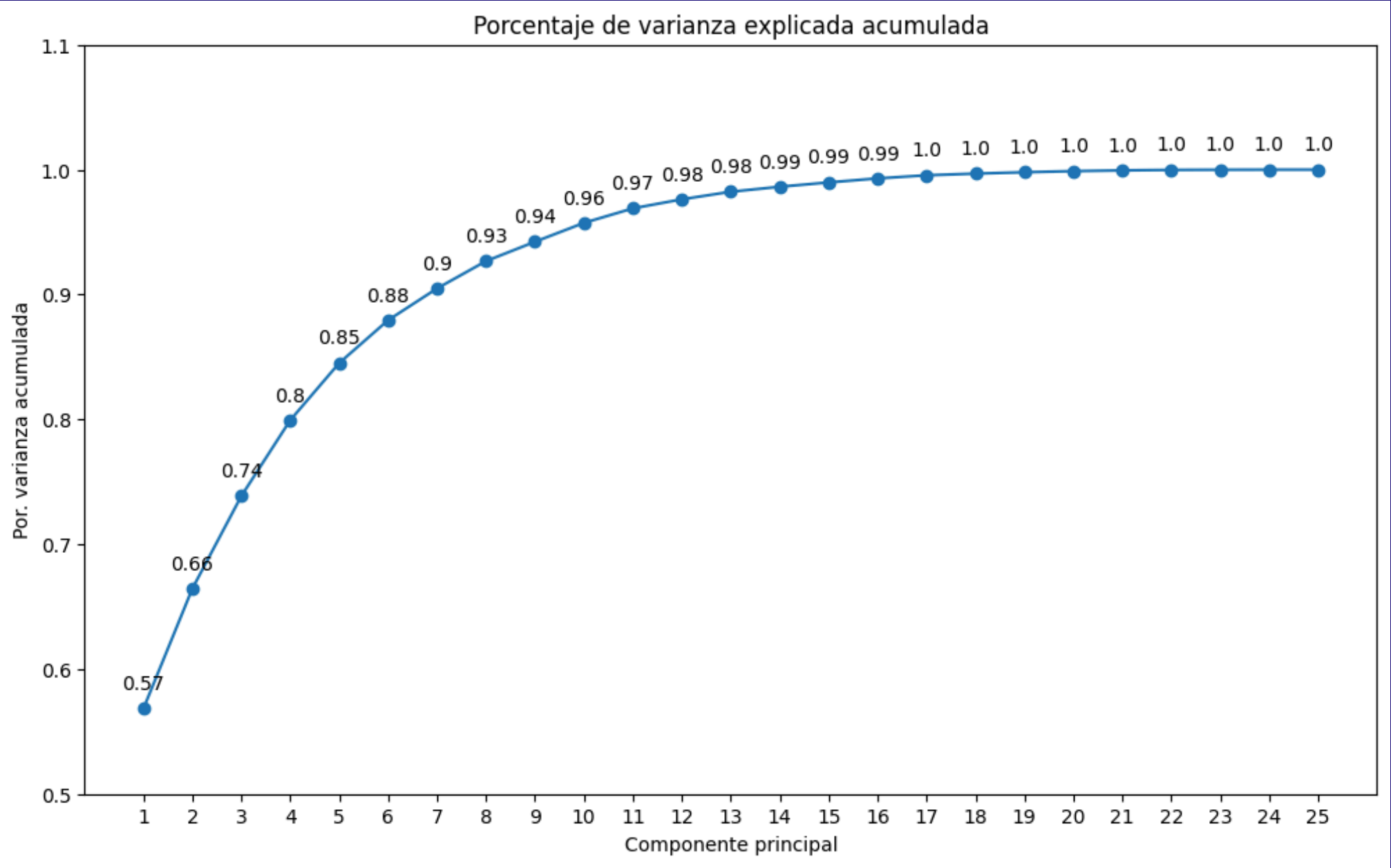


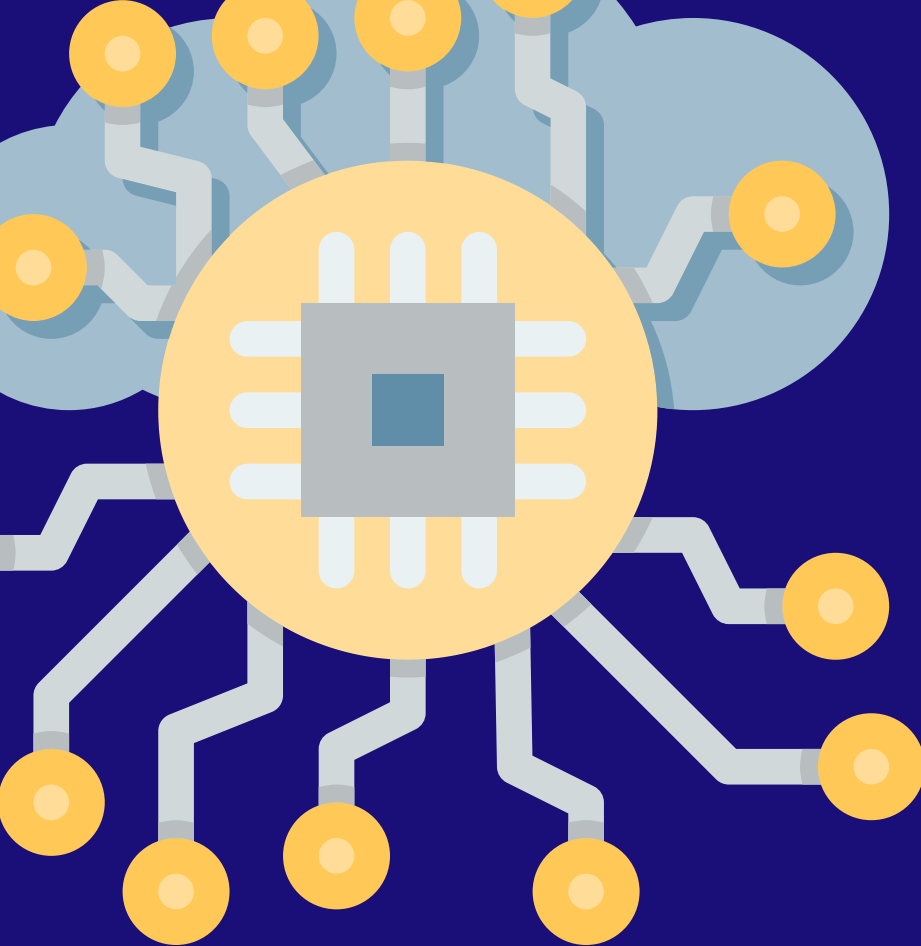
- classifier: RandomForestClassifier
- classifier\_\_max\_depth: 5
- classifier\_\_max\_leaf\_nodes: 18
- classifier\_\_n\_estimators: 500
- pca\_\_n\_components: 25
- scaler: StandardScaler()

	Feature	Component	Explained Variance Ratio
0	Time	PC1	0.568221
1	Location_encoded	PC2	0.096179
2	Zone_encoded	PC3	0.074115
3	Habitat_encoded	PC4	0.060819
4	Cranial_Capacity	PC5	0.045753
5	Height	PC6	0.034354
6	Incisor_Size_encoded	PC7	0.025444
7	Jaw_Shape_encoded	PC8	0.021881
8	Torus_Supraorbital_encoded	PC9	0.015482
9	Prognathism_encoded	PC10	0.015112
10	Foramen_encoded	PC11	0.011615
11	Canine_Size_encoded	PC12	0.007255
12	Canines_Shape_encoded	PC13	0.006075
13	Tooth_Enamel_encoded	PC14	0.003987
14	Tecno_encoded	PC15	0.003488
15	tecno_type_mapping_encoded	PC16	0.003212
16	Biped_encoded	PC17	0.002488
17	foots_encoded	PC18	0.001356
18	arms_encoded	PC19	0.001083
19	Diet_encoded	PC20	0.000897
20	Sexual_Dimorphism_encoded	PC21	0.000612
21	Hip_encoded	PC22	0.000355
22	Vertical_Front_encoded	PC23	0.000139
23	Anatomy_encoded	PC24	0.000070
24	Migrated_encoded	PC25	0.000008



	Feature	Component	Explained Variance Ratio
0	Time	PC1	0.568221
1	Location_encoded	PC2	0.096179
2	Zone_encoded	PC3	0.074115
3	Habitat_encoded	PC4	0.060819
4	Cranial_Capacity	PC5	0.045753
5	Height	PC6	0.034354
6	Incisor_Size_encoded	PC7	0.025444
7	Jaw_Shape_encoded	PC8	0.021881
8	Torus_Supraorbital_encoded	PC9	0.015482
9	Prognathism_encoded	PC10	0.015112
10	Foramen_encoded	PC11	0.011615
11	Canine_Size_encoded	PC12	0.007255
12	Canines_Shape_encoded	PC13	0.006075
13	Tooth_Enamel_encoded	PC14	0.003987
14	Tecno_encoded	PC15	0.003488
15	tecno_type_mapping_encoded	PC16	0.003212
16	Biped_encoded	PC17	0.002488
17	foots_encoded	PC18	0.001356
18	arms_encoded	PC19	0.001083
19	Diet_encoded	PC20	0.000897
20	Sexual_Dimorphism_encoded	PC21	0.000612
21	Hip_encoded	PC22	0.000355
22	Vertical_Front_encoded	PC23	0.000139
23	Anatomy_encoded	PC24	0.000070
24	Migrated_encoded	PC25	0.000008





## 6. EVALUACIÓN DEL MODELO

- **Precision:** de todas las instancias clasificadas como positivas, ¿cuántas realmente lo son?
- **Recall:** De todas las instancias que son realmente positivas, ¿Cuántas fueron identificadas correctamente por el modelo?
- **Accuracy:** Es la proporción de todas las predicciones que son correctas
- **F1-score:** Métrica de evaluación de modelos de clasificación que combina la precisión y el recall en un solo valor.



## MÉTRICAS CON TODAS LAS VARIABLES:

```
TEST
Precision: 1.0
Recall: 1.0
F1-score: 1.0
ROC AUC score: 1.0
Accuracy 1.0
```

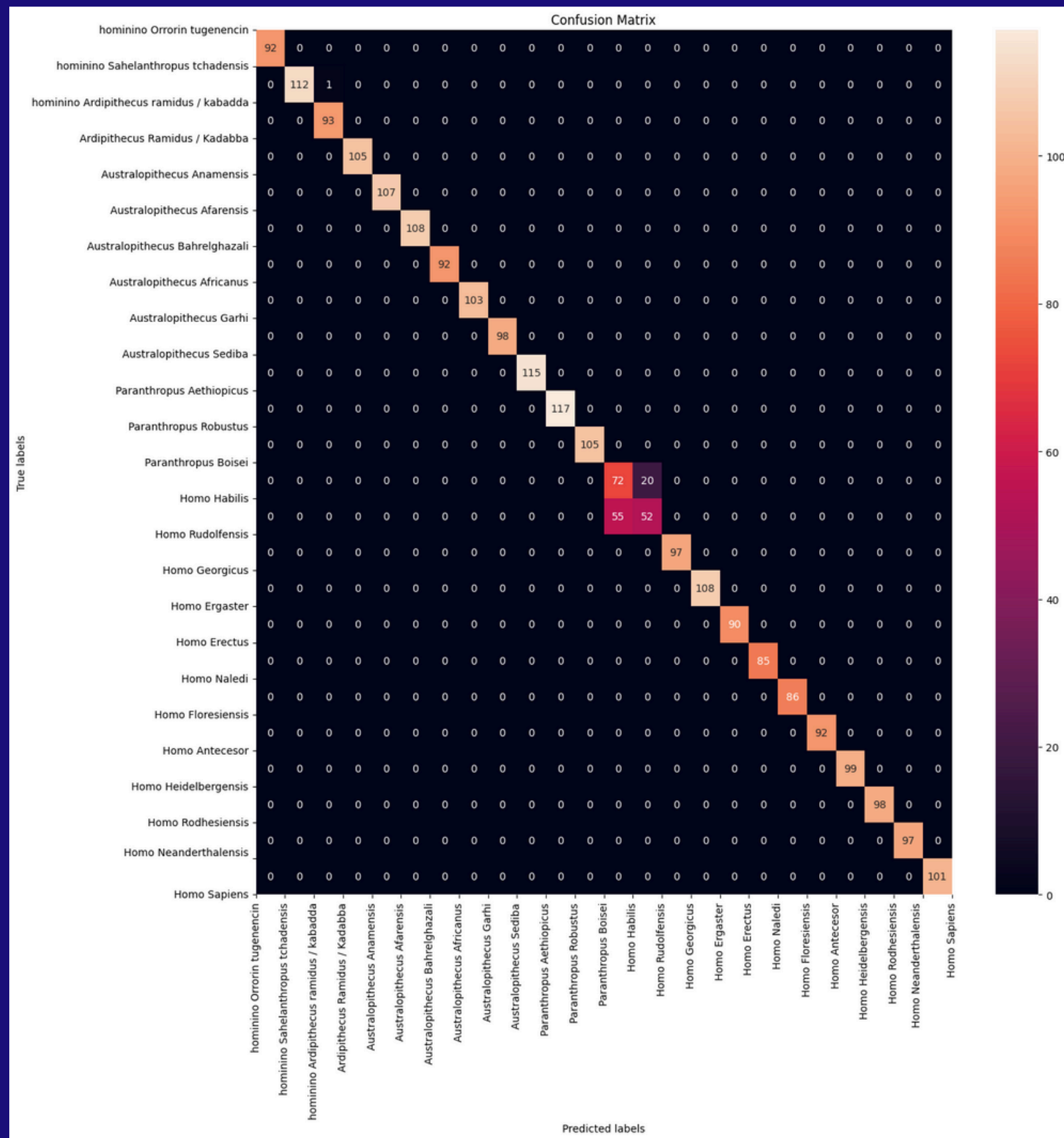
```
-----
TRAIN
Precision: 1.0
Recall: 1.0
F1-score: 1.0
ROC AUC score: 1.0
Accuracy 1.0
```

VS

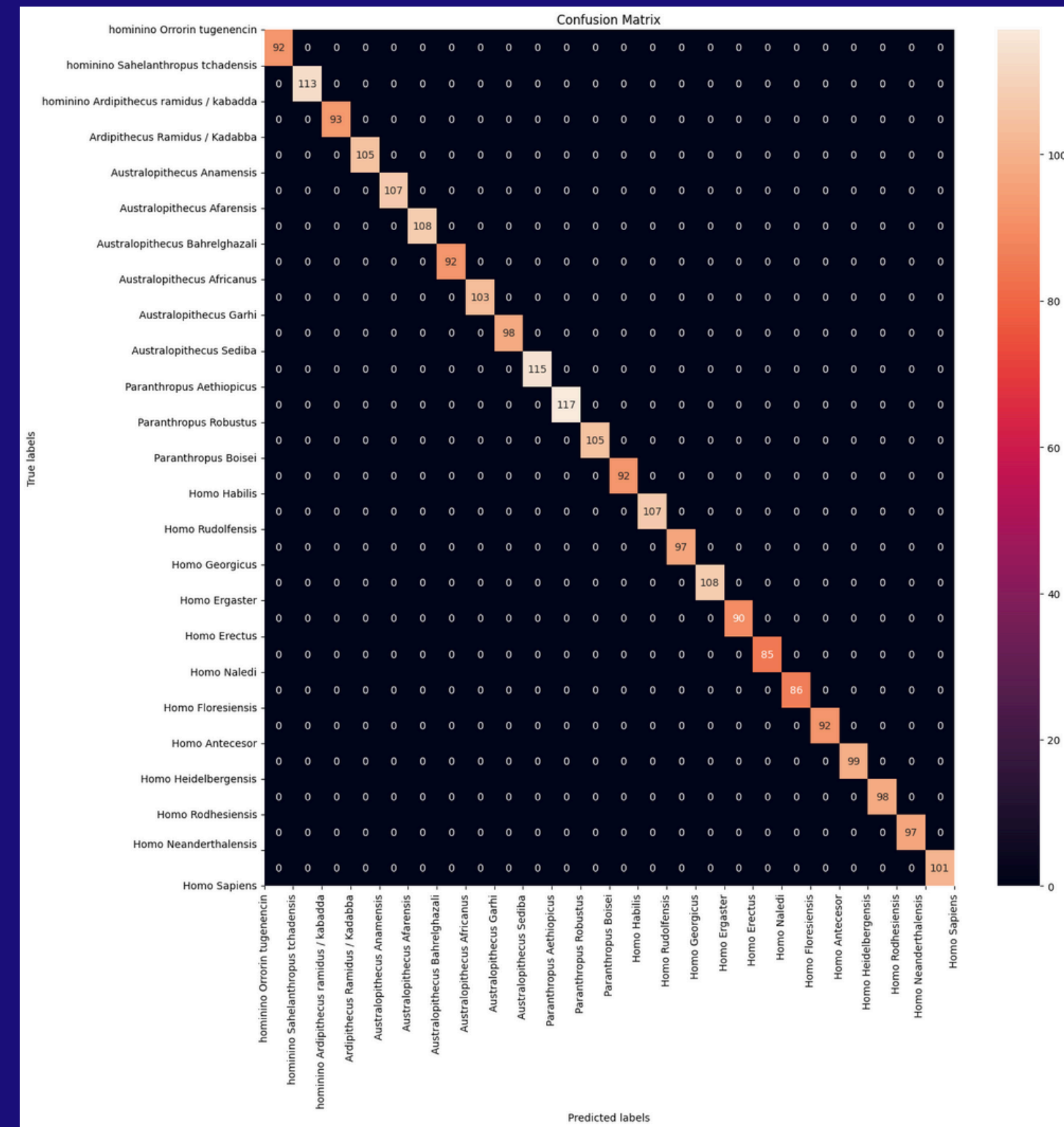
## METRICAS CON 6 COMPONENTES PRINCIPALES:

```
TEST
Precision: 0.9706024568294211
Recall: 0.9683333333333334
F1-score: 0.96777550001036
Accuracy 0.9683333333333334
```

```
-----
TRAIN
Precision: 0.9709938735659144
Recall: 0.9698958333333333
F1-score: 0.9690546916858926
Accuracy 0.9698958333333333
```



Modelo con PCA 6



Modelo con PCA 24



## 7.CONCLUSIONES

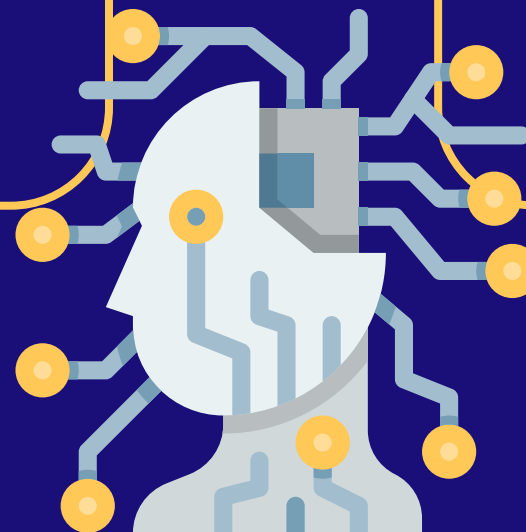
### ACIERTOS

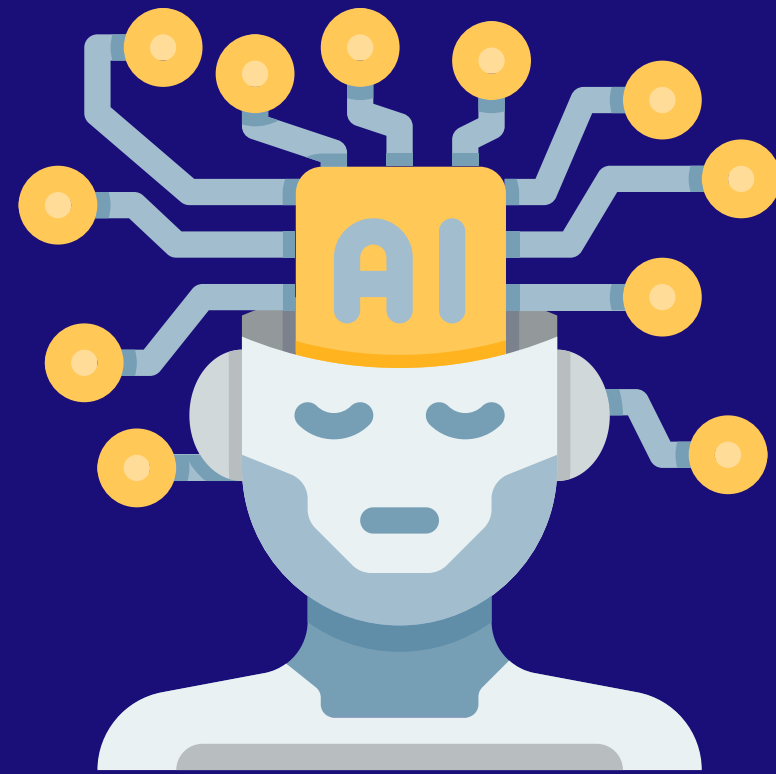
- El modelo ha conseguido unas métricas del 100% de aciertos en test.
- Es un modelo que no es muy costoso computacionalmente.
- Se puede seguir actualizando con nuevas variables fácilmente.
- Con pocas variables en función del PCA obtiene también muy buenas métricas.
- Es un modelo fácilmente interpretable

VS

### CONTRAS

- Los datos de origen deben venir bien clasificados y estructurados para que el modelo siga teniendo buenos rendimientos
- Habría que medir su rendimiento ante nuevos fósiles con características distintas a los datos de entrenamiento ya que se basa en datos muy deterministas.





# Gracias

Teresa Terol Díez  
<http://localhost:8501/>

[https://github.com/svalencia-romero/taller\\_opencv\\_thebridge\\_09\\_23](https://github.com/svalencia-romero/taller_opencv_thebridge_09_23)