A2.2 LDA y árboles de decisión

Luis Enrique Garcia Gallegos

Matricula: 649247

En esta actividad trabajarás con la misma base de datos que trabajaste en el proyecto del primer parcial y en la actividad A2.1. Si por algún motivo tienes problemas y deseas cambiar de base de datos, puedes hacerlo, especificando claramente con qué base de datos estás trabajando ahora y de dónde la obtuviste. Desarrolla los siguientes puntos en una *Jupyter Notebook*, tratando, dentro de lo posible, que cada punto se trabaje en una celda distinta. Los comentarios en el código siempre son bienvenidos, de preferencia, aprovecha el *markdown* para generar cuadros de descripción que ayuden al lector a comprender el trabajo realizado.

1. Importa los datos a tu ambiente de trabajo y separa los datos en entrenamiento y prueba, con una relación que consideres adecuada, manteniendo un balance de clases. Demuestra que se cumplió la condición imprimiendo datos relevantes en la consola.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
import random
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split, LeaveOneOut, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
BaseDeDatos=pd.read_csv('INE_SECCION_2020.csv')
datos=BaseDeDatos.drop(columns=['ID', 'DISTRITO', 'MUNICIPIO', 'SECCION', 'POBFEM', 'POBMAS', 'VPH_SINTIC', 'VPH_SINCIN', 'VPH
for i in range(32, 1, -1):
    name='ENTIDAD'+str(i)
    datos[name]=datos['ENTIDAD']//i
    existe=datos[name]*i
    datos['ENTIDAD']=datos['ENTIDAD']-existe
datos['TIPO']=datos['TIPO']-2
datos=datos.drop('ENTIDAD', axis=1)
X=datos.drop('TIPO', axis=1)
Y=datos['TIPO']
random.seed(0)
x_train, x_test, y_train, y_test=train_test_split(X, Y, test_size=0.2, stratify=Y)
print("Datos de entrenamiento: ", x_train.shape, "\tDatos de prueba: ", x_test.shape, "\n\tTotal de datos: ", (x_train.shape[0
print("\tSalida de entrenamiento (porcentajes):\n", round(100*(y_train.value_counts())/x_train.shape[0], 2), "\n\n\tSalida de
```

```
Datos de entrenamiento:  (55044, 46)    Datos de prueba:  (13762, 46)
        Total de datos:  68806
        Salida de entrenamiento (porcentajes):
 TIPO
0    63.68
2    27.04
1     9.28
Name: count, dtype: float64

        Salida de prueba (porcentajes):
 TIPO
0    63.68
2    27.04
1     9.28
Name: count, dtype: float64
```

Se realizo la limpieza de datos al igual que en los trabajos pasados, con el fin fe realizar un modelo que prediga clases como en el trabajo pasado, sin embargo esta vez usaremos otro modelo, ademas de evitar el desbalance de datos mediante la adición de `stratify=Y`, la cual hará que las particiones tengan misma proporción de datos; igualmente se uso una semilla.

2. Corre una regresión logística simple con la función `GLM` de `statsmodels` con todas tus variables de entrada e imprime el resumen del modelo en consola. Indica cuáles son las **2 variables** más relevantes para este estudio y selecciónalas, eliminando todas las demás variables de tus datos de entrenamiento y prueba.

In [2]:
```
x_train_fit=sm.add_constant(x_train)
x_test_fit=sm.add_constant(x_test)
modeloGLM=sm.MNLogit(y_train, x_train_fit)
resultadoGLM=modeloGLM.fit()
print(resultadoGLM.summary())
pValuesGLM=resultadoGLM.pvalues
print("\nValores P>|z| de cada variable (menor a mayor):\n", pValuesGLM.stack().sort_values())
y_pred_prob=resultadoGLM.predict(x_test_fit)
y_pred_lda=np.argmax(y_pred_prob.values, axis=1)
x_train=x_train[['P18YM_PB', 'P15PRI_IN']]
x_test=x_test[['P18YM_PB', 'P15PRI_IN']]
```

```
Optimization terminated successfully.
         Current function value: 0.502874
         Iterations 12
                          MNLogit Regression Results
==============================================================================
Dep. Variable:                   TIPO   No. Observations:                55044
Model:                        MNLogit   Df Residuals:                    54950
Method:                           MLE   Df Model:                           92
Date:                Thu, 20 Mar 2025   Pseudo R-squ.:                 0.4164
Time:                        02:19:03   Log-Likelihood:                -27680.
converged:                       True   LL-Null:                       -47430.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
    TIPO=1       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.8416      0.172    -16.518      0.000      -3.179      -2.504
POBTOT         0.0010      0.000      5.565      0.000       0.001       0.001
P_0A2          0.0076      0.001      9.649      0.000       0.006       0.009
P3A5_NOA      -0.0122      0.001    -10.842      0.000      -0.014      -0.010
P6A11_NOA      0.0037      0.002      1.486      0.137      -0.001       0.009
P12A14NOA      0.0021      0.003      0.701      0.483      -0.004       0.008
P15A17A        0.0091      0.001      9.728      0.000       0.007       0.011
P18A24A       -0.0155      0.001    -20.012      0.000      -0.017      -0.014
P8A14AN       -0.0129      0.003     -3.908      0.000      -0.019      -0.006
P15YM_AN       0.0055      0.001      6.558      0.000       0.004       0.007
P15YM_SE      -0.0041      0.001     -5.412      0.000      -0.006      -0.003
P15PRI_IN      0.0096      0.000     22.288      0.000       0.009       0.010
P15PRI_CO     -0.0040      0.000    -11.820      0.000      -0.005      -0.003
P15SEC_IN     -0.0152      0.001    -18.079      0.000      -0.017      -0.014
P15SEC_CO     -0.0006      0.000     -2.282      0.022      -0.001   -8.23e-05
P18YM_PB      -0.0010      0.000     -4.318      0.000      -0.001      -0.001
ENTIDAD32      1.5729      0.189      8.304      0.000       1.202       1.944
ENTIDAD31     -0.0847      0.206     -0.411      0.681      -0.488       0.319
ENTIDAD30      0.0235      0.185      0.127      0.899      -0.339       0.386
ENTIDAD29      0.8782      0.215      4.093      0.000       0.458       1.299
ENTIDAD28      0.3755      0.192      1.951      0.051      -0.002       0.753
ENTIDAD27     -0.8786      0.237     -3.710      0.000      -1.343      -0.414
ENTIDAD26      0.2175      0.206      1.058      0.290      -0.186       0.620
ENTIDAD25      0.7442      0.183      4.076      0.000       0.386       1.102
ENTIDAD24      0.3720      0.207      1.798      0.072      -0.034       0.777
ENTIDAD23     -0.2314      0.223     -1.039      0.299      -0.668       0.205
ENTIDAD22      1.0896      0.209      5.221      0.000       0.681       1.499
ENTIDAD21      0.0673      0.190      0.354      0.723      -0.305       0.440
ENTIDAD20      1.1501      0.185      6.207      0.000       0.787       1.513
ENTIDAD19     -0.2098      0.194     -1.081      0.280      -0.590       0.171
ENTIDAD18      0.2248      0.219      1.024      0.306      -0.205       0.655
ENTIDAD17      0.5615      0.206      2.722      0.006       0.157       0.966
ENTIDAD16      0.4801      0.189      2.546      0.011       0.110       0.850
ENTIDAD15     -0.4592      0.185     -2.478      0.013      -0.822      -0.096
ENTIDAD14      0.4506      0.183      2.466      0.014       0.092       0.809
ENTIDAD13      1.3207      0.199      6.639      0.000       0.931       1.711
ENTIDAD12      0.2359      0.200      1.180      0.238      -0.156       0.628
ENTIDAD11     -0.1337      0.192     -0.697      0.486      -0.510       0.242
ENTIDAD10      0.4617      0.216      2.141      0.032       0.039       0.884
ENTIDAD9      -1.5317      0.233     -6.563      0.000      -1.989      -1.074
ENTIDAD8      -0.0123      0.193     -0.064      0.949      -0.391       0.366
ENTIDAD7      -0.8250      0.210     -3.924      0.000      -1.237      -0.413
ENTIDAD6       1.2867      0.233      5.520      0.000       0.830       1.744
ENTIDAD5       0.0849      0.202      0.419      0.675      -0.312       0.482
ENTIDAD4       0.2969      0.252      1.176      0.239      -0.198       0.792
ENTIDAD3       0.4822      0.246      1.957      0.050      -0.001       0.965
ENTIDAD2      -0.4752      0.227     -2.090      0.037      -0.921      -0.030
------------------------------------------------------------------------------
    TIPO=2       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.3932      0.174     -2.256      0.024      -0.735      -0.052
POBTOT         0.0018      0.000      6.218      0.000       0.001       0.002
P_0A2          0.0177      0.001     13.984      0.000       0.015       0.020
P3A5_NOA      -0.0137      0.001     -9.541      0.000      -0.017      -0.011
P6A11_NOA      0.0138      0.003      4.936      0.000       0.008       0.019
P12A14NOA     -0.0283      0.003     -8.714      0.000      -0.035      -0.022
P15A17A        0.0155      0.001     10.914      0.000       0.013       0.018
P18A24A       -0.0081      0.001     -5.436      0.000      -0.011      -0.005
P8A14AN       -0.0007      0.003     -0.244      0.807      -0.007       0.005
P15YM_AN       0.0049      0.001      5.614      0.000       0.003       0.007
P15YM_SE      -0.0070      0.001     -8.603      0.000      -0.009      -0.005
P15PRI_IN      0.0090      0.000     18.243      0.000       0.008       0.010
P15PRI_CO     -0.0061      0.000    -13.668      0.000      -0.007      -0.005
P15SEC_IN     -0.0187      0.001    -16.254      0.000      -0.021      -0.016
P15SEC_CO     -0.0009      0.000     -2.230      0.026      -0.002      -0.000
P18YM_PB      -0.0124      0.000    -33.129      0.000      -0.013      -0.012
ENTIDAD32      1.0472      0.186      5.640      0.000       0.683       1.411
ENTIDAD31     -0.9117      0.219     -4.155      0.000      -1.342      -0.482
ENTIDAD30      1.3591      0.182      7.472      0.000       1.003       1.716
ENTIDAD29      0.6654      0.227      2.931      0.003       0.220       1.110
ENTIDAD28      0.5095      0.192      2.648      0.008       0.132       0.887
ENTIDAD27      2.2573      0.206     10.942      0.000       1.853       2.662
```

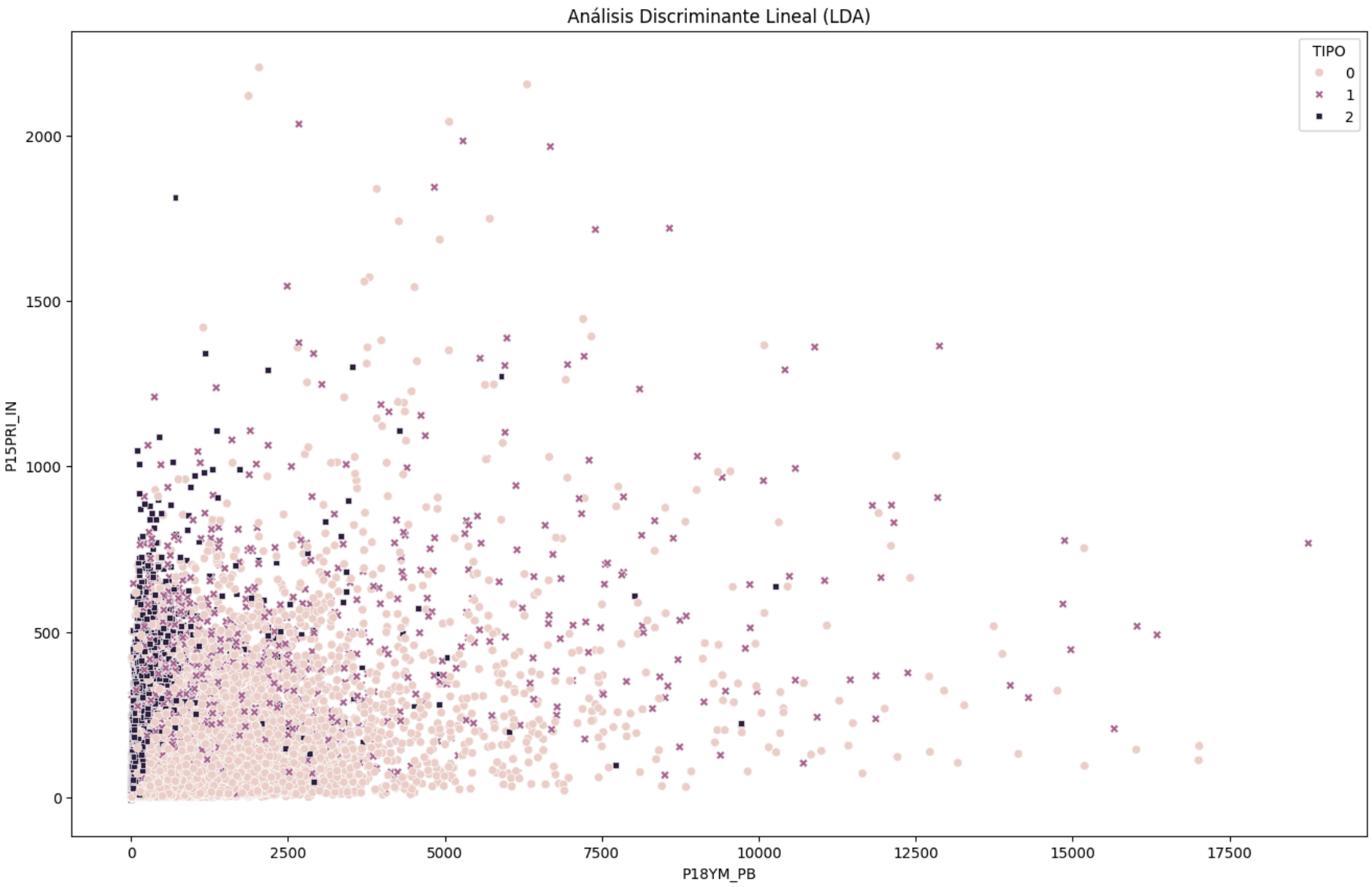
```
ENTIDAD26      1.0259      0.197      5.212      0.000       0.640       1.412
ENTIDAD25      0.4151      0.180      2.309      0.021       0.063       0.767
ENTIDAD24      1.6761      0.192      8.717      0.000       1.299       2.053
ENTIDAD23     -0.4180      0.226     -1.853      0.064      -0.860       0.024
ENTIDAD22      1.4258      0.223      6.387      0.000       0.988       1.863
ENTIDAD21      0.5220      0.190      2.745      0.006       0.149       0.895
ENTIDAD20      0.5226      0.188      2.787      0.005       0.155       0.890
ENTIDAD19      0.0309      0.190      0.162      0.871      -0.342       0.404
ENTIDAD18      0.9581      0.197      4.866      0.000       0.572       1.344
ENTIDAD17      0.8748      0.229      3.817      0.000       0.426       1.324
ENTIDAD16      0.9079      0.185      4.900      0.000       0.545       1.271
ENTIDAD15      1.8622      0.183     10.183      0.000       1.504       2.221
ENTIDAD14      0.7851      0.185      4.238      0.000       0.422       1.148
ENTIDAD13      3.0560      0.198     15.466      0.000       2.669       3.443
ENTIDAD12      1.7771      0.188      9.476      0.000       1.410       2.145
ENTIDAD11      0.8001      0.182      4.398      0.000       0.444       1.157
ENTIDAD10      1.4637      0.192      7.640      0.000       1.088       1.839
ENTIDAD9      -3.6372      1.018     -3.575      0.000      -5.632      -1.643
ENTIDAD8       0.1502      0.182      0.824      0.410      -0.207       0.507
ENTIDAD7       1.2036      0.197      6.097      0.000       0.817       1.591
ENTIDAD6       0.8032      0.268      3.002      0.003       0.279       1.328
ENTIDAD5       0.3938      0.192      2.047      0.041       0.017       0.771
ENTIDAD4       1.4190      0.231      6.136      0.000       0.966       1.872
ENTIDAD3       0.9790      0.243      4.033      0.000       0.503       1.455
ENTIDAD2       0.1754      0.208      0.843      0.399      -0.232       0.583
==============================================================================

Valores P>|z| de cada variable (menor a mayor):
 P18YM_PB   1    1.121795e-240
P15PRI_IN  0    4.785292e-110
P18A24A    0     4.365562e-89
P15PRI_IN  1     2.356563e-74
P15SEC_IN  0     4.671962e-73
                     ...
ENTIDAD21  0     7.231935e-01
P8A14AN    1     8.071493e-01
ENTIDAD19  1     8.711511e-01
ENTIDAD30  0     8.988994e-01
ENTIDAD8   0     9.490603e-01
Length: 94, dtype: float64
```

Para poder escoger las variables más relevantes se uso `summary()` para ver cuales variables tienen un *p-value* de menor valor, como más de una variables tienen un *p-value* menor de **0.0001** se imprimio los *p-values* de cada variable (menor a mayor), con el fin de escoger las **2 variables** más relevantes, siendo `P18YM_PB` y `P15PRI_IN`. Una cosa a mencionar es que como tal no se corrio una `GLM` debido a que esta es para predecir una clase binaria, como estamos prediciendo una multiclase entonces usamos la `MNLogit`.

3. Genera un modelo usando la metodología de linear discriminant analysis. Visualiza la función discriminante con una gráfica de variable 1 vs variable 2, donde cada observación tenga algo que la distinga dependiendo de su clase (por ejemplo, distinto color, distinto marcador, etc.)

In [3]:
```
modeloLDA=LinearDiscriminantAnalysis()
resultadoGLDA=modeloLDA.fit(x_train, y_train)
plt.figure(figsize=(16, 10))
sns.scatterplot(x=x_train['P18YM_PB'], y=x_train['P15PRI_IN'], hue=y_train, style=y_train)
plt.title('Análisis Discriminante Lineal (LDA)')
plt.xlabel('P18YM_PB')
plt.ylabel('P15PRI_IN')
plt.legend(title="TIPO")
plt.show()
```

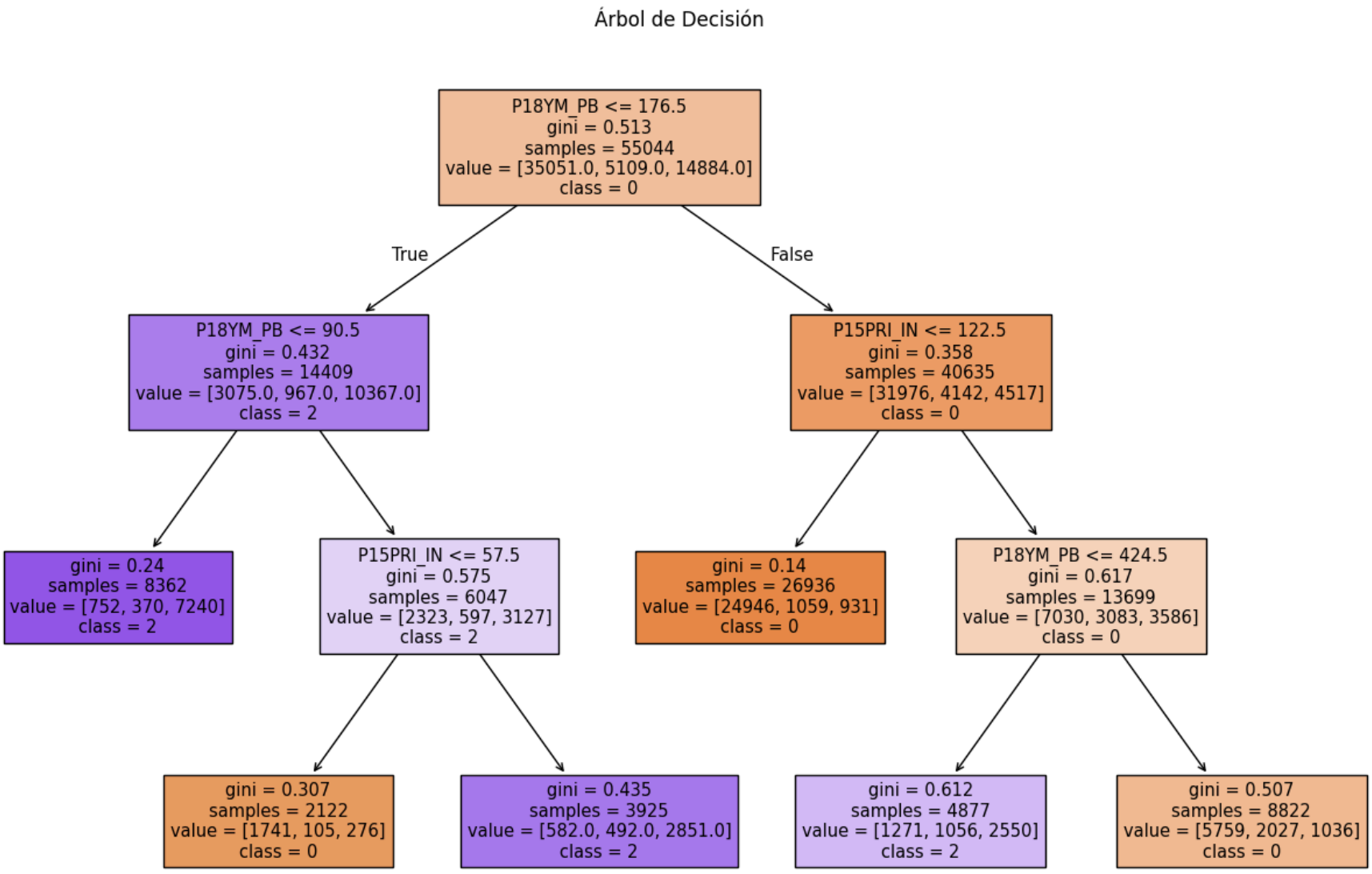


Chat-gpt proporciono parte del código el cual nos permitirá observar como se comportan nuestros datos según `LinearDiscriminantAnalysis()`, esto con el fin de poder general un árbol de decisiones y el poder las ramas que se formaran en nuestro modelo.

4. Genera un modelo usando la metodología de árboles de decisión. Deberás podar el árbol, habiendo seleccionado primero un valor óptimo de α mediante una metodología de LOOCV. Visualiza tanto el árbol resultante, como la partición en una gráfica de variable 1 vs variable 2, donde cada observación tenga algo que la distinga dependiendo de su clase.

In [4]:
```
arbol_base=DecisionTreeClassifier(random_state=42)
arbol_base.fit(x_train, y_train)
param_grid={'ccp_alpha': np.linspace(0, 0.1, 20)}
grid_search=GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, cv=5, scoring='accuracy')
grid_search.fit(x_train, y_train)
best_alpha=grid_search.best_params_['ccp_alpha']
print(f"Mejor α encontrado: {best_alpha}")
arbol_podado=DecisionTreeClassifier(random_state=42, ccp_alpha=best_alpha)
arbol_podado.fit(x_train, y_train)
plt.figure(figsize=(16, 10))
plot_tree(arbol_podado, filled=True, feature_names=['P18YM_PB', 'P15PRI_IN'], class_names=np.unique(y_train).astype(str))
plt.title("Árbol de Decisión")
plt.show()
```

```
Mejor α encontrado: 0.005263157894736842
```

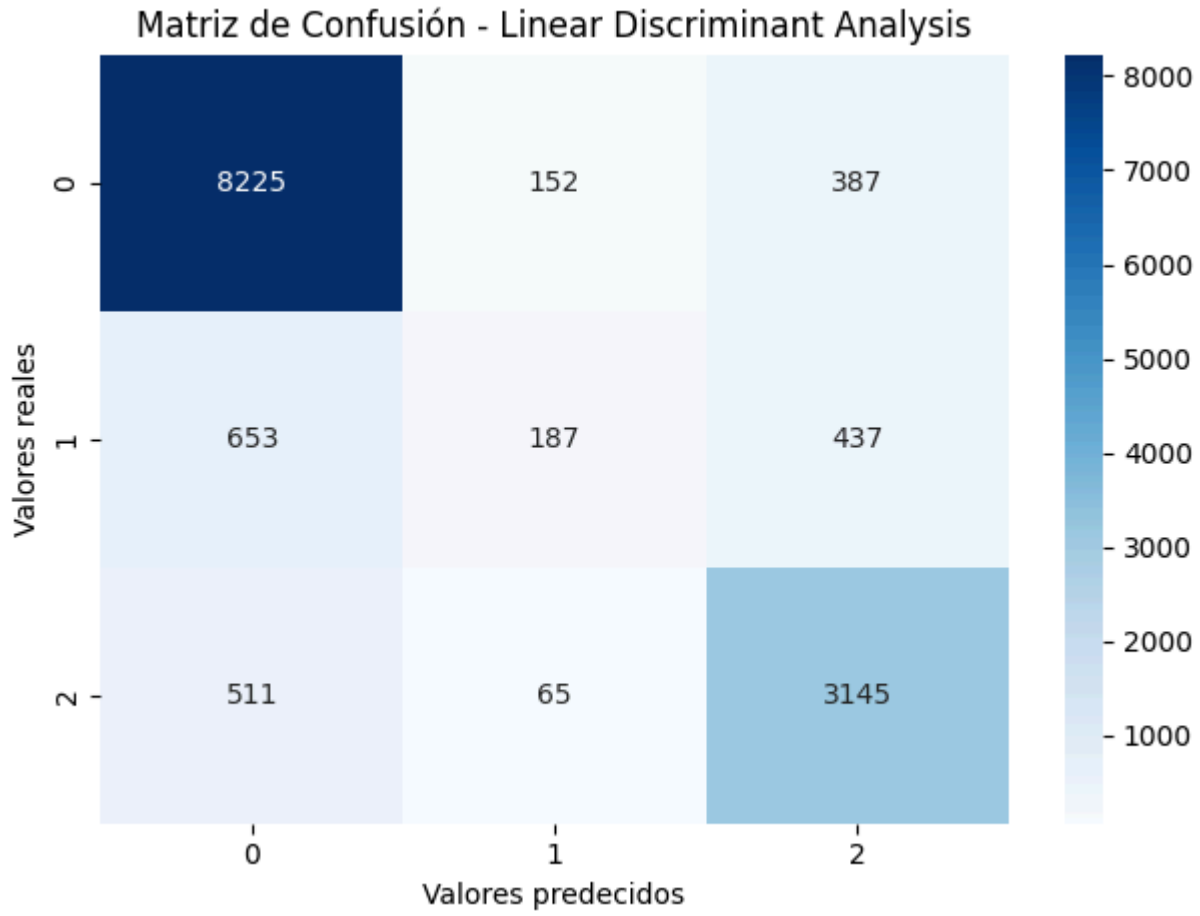


Este código fue proporcionado por Chat-gpt, no obstante se realizaron cambios para nuestro caso, el `random_state=42` nos dice la reproducibilidad de nuestro modelo de tal manera que evitamos caer en sesgos. Lo que hace el código es ejecutar varios modelos de arboles e ir ajustado sus parámetros para bajar el accuracy con el fin de evitar el sobre ajuste, a este proceso se le conoce como poda del árbol, se realizo varios cambios al código, debido a que tenemos un gran numero de datos ademas de que muchos de los parámetros que nos daba chat provocaban que se tardara mucho tiempo el poder conseguir el α que mejor se adaptara. Las ramas de nuestro árbol de decision ya podado es importante mencionar que este descarta la posibilidad de que tengamos **la clase 1**, por lo que unicamente nos dirá si algo pertenece a la clase **0** o **2**, esto se puede deber a la gráfica que se mostró antes, ya que si nos fijamos bien vemos que la **clase 1** esta muy dispersa, no obstante la **clase 0** se encuentra mas en una región.

5. Calcula, para ambos modelos, todas las métricas revisadas en clase en los datos de prueba. Indica qué opinas sobre los resultados, especificando si crees que uno de los dos modelos es mejor para esta tarea específica.

In [5]:
```
y_pred_arbol=arbol_podado.predict(x_test)
def evaluar_modelo(y_test, y_pred, nombre_modelo):
    print("\t", nombre_modelo)
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
    print(f"Precision: {precision_score(y_test, y_pred, average='weighted'):.4f}")
    print(f"Recall: {recall_score(y_test, y_pred, average='weighted'):.4f}")
    print(f"F1-Score: {f1_score(y_test, y_pred, average='weighted'):.4f}")
    cm=confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
    plt.ylabel('Valores reales')
    plt.xlabel('Valores predecidos')
    plt.title(f'Matriz de Confusión - {nombre_modelo}')
    plt.tight_layout()
    plt.show()
    print("\nReporte de clasificación:")
    print(classification_report(y_test, y_pred))
evaluar_modelo(y_test, y_pred_lda, "Linear Discriminant Analysis")
evaluar_modelo(y_test, y_pred_arbol, "Árbol de Decisión")
```

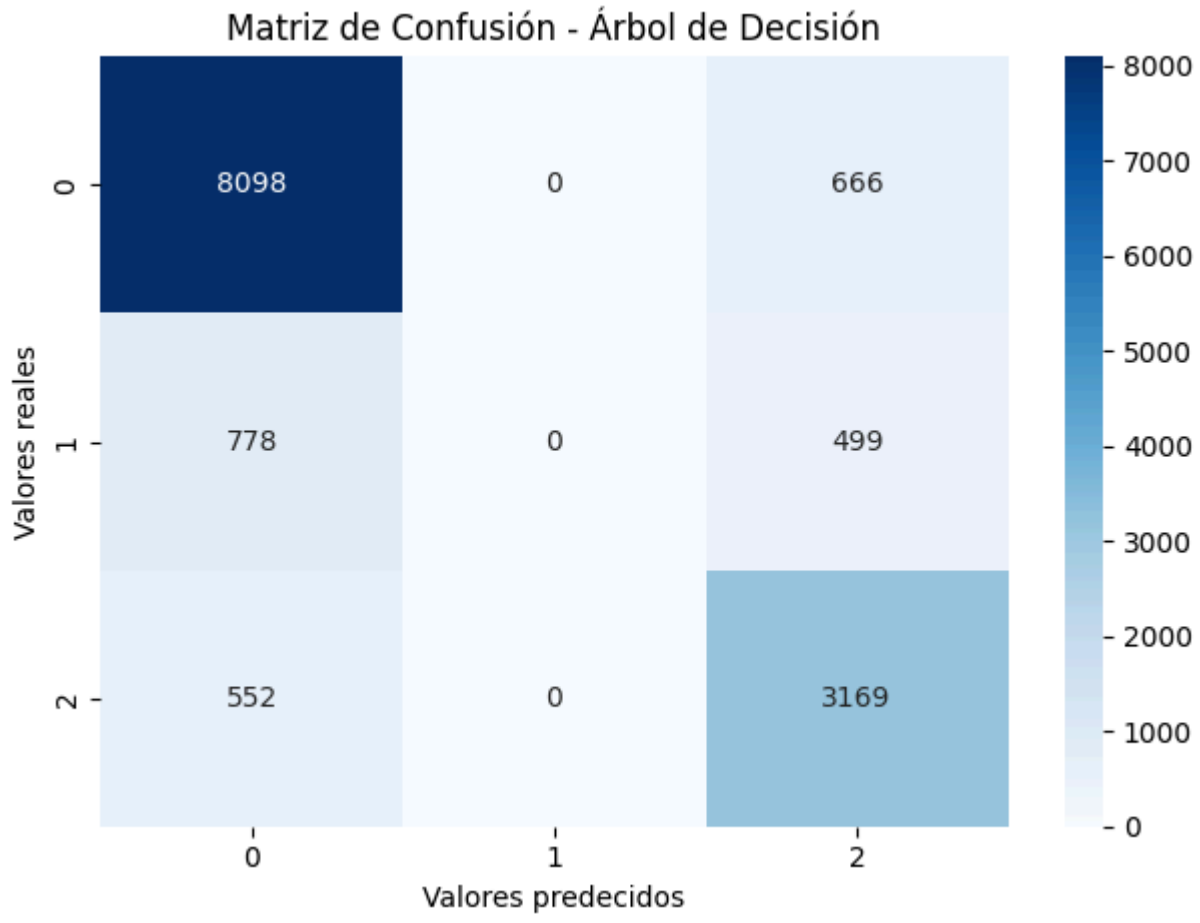
```
        Linear Discriminant Analysis
Accuracy: 0.8398
Precision: 0.8151
Recall: 0.8398
F1-Score: 0.8189
```



Reporte de clasificación:

	precision	recall	f1-score	support
0	0.88	0.94	0.91	8764
1	0.46	0.15	0.22	1277
2	0.79	0.85	0.82	3721
accuracy			0.84	13762
macro avg	0.71	0.64	0.65	13762
weighted avg	0.82	0.84	0.82	13762

Árbol de Decisión
Accuracy: 0.8187
Precision: 0.7447
Recall: 0.8187
F1-Score: 0.7797



Reporte de clasificación:

	precision	recall	f1-score	support
0	0.86	0.92	0.89	8764
1	0.00	0.00	0.00	1277
2	0.73	0.85	0.79	3721
accuracy			0.82	13762
macro avg	0.53	0.59	0.56	13762
weighted avg	0.74	0.82	0.78	13762

Este código fue proporcionado por Chat-gpt, vemos varias métricas y las matrices de confusion, pero en general nos podemos dar cuenta por los `Accuracy`, `Precision` y `F1-Score`, que el `LinearDiscriminantAnalysis` fue el que mejor se adapta a nuestros datos, no obstante el árbol de decision no se queda tan lejos de estos valores. Las matrices de confusion nos muestran que en general ambos modelos tiene

problemas para determinar si pertenece a la **clase 1**, esto se corrobora con la cantidad de datos que hay de esa clase, esto justamente se ve en cuanto nos fijamos en los `precision`, `recall` y `support`, los cuales muestran un claro desbalance de datos.

Firma de Honor: Doy mi palabra que he realizado esta actividad con integridad académica