El pipeline utilizado en este modelo sigue una secuencia de pasos que son comunes en el preprocesamiento y modelado de datos para clasificación. A continuación, se explica cada componente del pipeline y cómo se conecta con el flujo general de la solución:

**1. Preprocesamiento de los Datos:**

* StandardScaler: Esta etapa se encarga de normalizar las características numéricas para que tengan una media de 0 y una desviación estándar de 1. Esto es importante cuando se utilizan algoritmos sensibles a la escala de los datos, como los árboles de decisión o modelos lineales.
* OneHotEncoder: Codifica las características categóricas en formato binario (0 o 1), de manera que puedan ser utilizadas por el modelo. Este paso es esencial para convertir variables categóricas en un formato compatible con los algoritmos de aprendizaje automático.

**2. Modelo:**

* DecisionTreeClassifier: Este es el modelo principal utilizado para la clasificación. En este caso, se utilizó un árbol de decisión con un criterio de "entropía" y una profundidad máxima (max\_depth) de 5. Los árboles de decisión son intuitivos y ofrecen interpretabilidad, al dividir los datos en función de características relevantes hasta llegar a decisiones finales.

**3. Entrenamiento y Validación:**

* Durante el proceso de entrenamiento, el pipeline aplica las transformaciones (escalado y codificación) a los datos y luego entrena el modelo de clasificación. Este flujo integrado asegura que las mismas transformaciones se apliquen a los datos de prueba y a cualquier dato nuevo, manteniendo la coherencia del proceso.

**4. Evaluación del Modelo:**

* Matriz de confusión: Se observa que el modelo tiene un alto desempeño en la clase mayoritaria (clase 0, que representa "no deserción") con 49,451 predicciones correctas, pero es menos preciso al clasificar la clase minoritaria (clase 1, "deserción"), con 3,835 falsos negativos.
* Reporte de Clasificación: El modelo alcanza una precisión global del 91%, con un desempeño equilibrado en términos de recall y f1-score para la clase minoritaria.
* ROC AUC: El área bajo la curva ROC de 0.92 indica un buen rendimiento global del modelo en distinguir entre clases.

**5. Importancia de Características:**

* El modelo identificó como más importantes características como CICLO\_RELATIVO, EDAD, EDAD\_INICIO\_UNI, y las calificaciones en el CEPRE. Estas variables influyen fuertemente en la predicción de deserción, indicando que factores como el ciclo de estudios y la edad relativa son claves en este contexto.

**Comparación con otras metodologías:**

* Si bien se podría haber optado por modelos más complejos como Random Forests o modelos de ensamble, se eligió un árbol de decisión por su capacidad de interpretabilidad. Aunque estos modelos más avanzados podrían mejorar ligeramente el rendimiento, el árbol de decisión brinda una ventaja en la transparencia y facilidad para explicar las decisiones.

**Decisión Final:**

* La elección de este pipeline fue motivada por un equilibrio entre la precisión del modelo, su interpretabilidad y la simplicidad para implementar un flujo reproducible y escalable. Las métricas obtenidas validan que el modelo es efectivo en la tarea, aunque se pueden explorar ajustes adicionales para mejorar la detección de la clase minoritaria.

**Resumen Explicativo del Modelo Utilizado**

**1. Modelo Utilizado**

Se implementó un modelo de **árbol de decisión** con el criterio de entropía y una profundidad máxima de 5 niveles. Este enfoque fue seleccionado por su capacidad de interpretar fácilmente las reglas de clasificación, lo que permite visualizar cómo se toman decisiones en función de las características más importantes. Además, se aplicó un pipeline con técnicas de preprocesamiento como escalado estándar para variables numéricas y codificación one-hot para variables categóricas.

**2. Lógica de la Decisión**

El modelo prioriza la importancia de ciertas características para predecir la deserción estudiantil. Según los resultados, las variables más relevantes fueron:

* **CICLO\_RELATIVO** (38.1%)
* **EDAD** (16.1%)
* **EDAD\_INICIO\_UNI** (12.4%)
* **Calificaciones en CEPRE (mínima, máxima y promedio)** y **domicilio por macroregión**

Estas variables dominan el proceso de clasificación, sugiriendo que el momento en que un estudiante inicia o avanza en su carrera, su edad y su desempeño académico previo son determinantes para predecir si persistirá o desertará.

**3. Comparativa con Otras Metodologías**

El árbol de decisión fue comparado con otros modelos como regresión logística y SVM, pero se optó por el árbol debido a su balance entre interpretabilidad y precisión. Modelos más complejos como los bosques aleatorios podrían ofrecer mayor precisión, pero a costa de la interpretabilidad.

**4. Resultados y Métricas de Evaluación**

El desempeño del modelo fue evaluado con las siguientes métricas:

* **Matriz de Confusión:**
  + Verdaderos Negativos: 49,451
  + Falsos Positivos: 3,835
  + Falsos Negativos: 1,327
  + Verdaderos Positivos: 3,104
* **Reporte de Clasificación:**
  + Precisión (Clase 0): 97% (baja deserción)
  + Precisión (Clase 1): 45% (alta deserción)
  + Recall (Clase 1): 70% (capacidad de identificar desertores)
  + F1-Score (Clase 1): 55%
* **Curva ROC (Área bajo la curva):** 0.92

El modelo muestra un buen desempeño general (91% de precisión), pero su precisión para la clase 1 (desertores) es moderada. El recall elevado para la clase 1 sugiere que el modelo identifica a la mayoría de los desertores, aunque con algunos falsos positivos.

**5. Decisión Final e Interpretación**

Aunque la precisión en la clase 1 es moderada, el objetivo principal es minimizar la deserción, por lo que se prioriza un alto recall para detectar posibles desertores. El modelo logra este objetivo al identificar correctamente el 70% de los casos de deserción, lo que es crucial para diseñar intervenciones preventivas. La importancia de las variables sugiere que el análisis de los ciclos y la edad pueden ser los principales indicadores de riesgo de deserción, permitiendo focalizar recursos en los estudiantes más vulnerables.

En resumen, el modelo es útil y proporciona insights clave para la toma de decisiones estratégicas, con margen de mejora en la reducción de falsos positivos si se implementan modelos más avanzados o combinaciones de técnicas de ensamble.

*import* pandas *as* pd

*import* numpy *as* np

*from* sklearn.model\_selection *import* train\_test\_split

*from* sklearn.pipeline *import* Pipeline

*from* sklearn.compose *import* ColumnTransformer

*from* sklearn.impute *import* SimpleImputer

*from* sklearn.preprocessing *import* StandardScaler, OneHotEncoder

*from* sklearn.ensemble *import* RandomForestClassifier

*from* imblearn.pipeline *import* Pipeline *as* ImbPipeline

*from* imblearn.over\_sampling *import* SMOTE

*from* sklearn.metrics *import* confusion\_matrix, classification\_report, roc\_curve, auc

*import* matplotlib.pyplot *as* plt

*import* seaborn *as* sns

rutafin = 'C:/Users/omarc/OneDrive/Escritorio/Proyectos VSC/OmarCano/Track Analítico UNI/05\_Datos/03\_Datos\_Finales'

tablon\_final\_df = pd.read\_csv(f'{rutafin}/tablon\_final\_df.csv')

df\_modelo = tablon\_final\_df[tablon\_final\_df['FLG\_ESTUDIANTE'] == 1].copy()

# *Imputación de valores nulos*

df\_modelo['DOMICILIO\_DEPA'] = df\_modelo['DOMICILIO\_DEPA'].fillna('DESCONOCIDO').astype(str).str.upper()

df\_modelo['NACIMIENTO\_DEPA'] = df\_modelo['NACIMIENTO\_DEPA'].replace('0', 'DESCONOCIDO').astype(str).str.upper()

df\_modelo['NACIMIENTO\_PAIS'] = df\_modelo['NACIMIENTO\_PAIS'].replace('0', 'DESCONOCIDO').astype(str).str.upper()

df\_modelo['COLEGIO\_PAIS'] = df\_modelo['COLEGIO\_PAIS'].replace('0', 'DESCONOCIDO').astype(str).str.upper()

df\_modelo['COLEGIO\_DEPA'] = df\_modelo['COLEGIO\_DEPA'].replace('0', 'DESCONOCIDO').astype(str).str.upper()

# *Agrupación de departamentos por macroregión*

lima = ['LIMA']

callao = ['CALLAO']

costa = ['PIURA', 'TUMBES','LAMBAYEQUE','LA LIBERTAD','ÁNCASH','ANCASH','ICA','MOQUEGUA', 'AREQUIPA','TACNA']

sierra = ['CAJAMARCA','HUÁNUCO','HUANUCO','PASCO', 'JUNIN', 'JUNÍN', 'HUANCAVELICA', 'AYACUCHO','APURIMAC','APURÍMAC',

          'CUSCO', 'PUNO']

selva = ['AMAZONAS', 'SAN MARTÍN', 'UCAYALI', 'MADRE DE DIOS', 'LORETO']

def *clasificar\_macroregion*(depa):

*if* depa in lima:

*return* 'Lima'

*elif* depa in callao:

*return* 'Callao'

*elif* depa in costa:

*return* 'Costa'

*elif* depa in sierra:

*return* 'Sierra'

*elif* depa in selva:

*return* 'Selva'

*else*:

*return* 'Desconocido'

df\_modelo['DOMICILIO\_MACROREGION'] = df\_modelo['DOMICILIO\_DEPA'].apply(clasificar\_macroregion)

df\_modelo['NACIMIENTO\_MACROREGION'] = df\_modelo['NACIMIENTO\_DEPA'].apply(clasificar\_macroregion)

df\_modelo['COLEGIO\_MACROREGION'] = df\_modelo['COLEGIO\_DEPA'].apply(clasificar\_macroregion)

# *Agrupación de países*

peru = ['PERÚ','PERU']

extranjero = ['ARGENTINA','VENEZUELA','JAPON','COREA REPUBLICA','ITALIA','ESTADOS UNIDOS','BRASIL','SUIZA','COLOMBIA','RSS DE UCRANIA',

              'PARAGUAY','CHILE','FRANCIA','NUEVA ZELANDA','BOLIVIA','RUSIA','GUATEMALA','ALEMANIA','ESPAÑA']

def *clasificar\_pais*(pais):

*if* pais in peru:

*return* 'Perú'

*elif* pais in extranjero:

*return* 'Extranjero'

*else*:

*return* 'Desconocido'

# *Aplicar la función a las columnas de país*

df\_modelo['NACIMIENTO\_PAIS'] = df\_modelo['NACIMIENTO\_PAIS'].apply(clasificar\_pais)

df\_modelo['COLEGIO\_PAIS'] = df\_modelo['COLEGIO\_PAIS'].apply(clasificar\_pais)

# *1. Definir las variables categóricas y numéricas*

categorical\_features = [

    'MODALIDAD\_AGRUP', 'DOMICILIO\_MACROREGION', 'NACIMIENTO\_MACROREGION', 'COLEGIO\_MACROREGION',

    'NACIMIENTO\_PAIS', 'COLEGIO\_PAIS'

]

numeric\_features = [

    'CICLO\_RELATIVO','EDAD','EDAD\_FIN\_COLEGIO','EDAD\_INICIO\_UNI','FLG\_CAMBIO\_CARRERA',

    'MIN\_CALIF\_ADMIS','MAX\_CALIF\_ADMIS','PROM\_CALIF\_ADMIS', 'CANT\_ADMIS',

    'MIN\_CALIF\_CEPRE', 'MAX\_CALIF\_CEPRE','PROM\_CALIF\_CEPRE', 'CANT\_CEPRE'

]

# *2. Preprocesamiento de datos*

preprocessor = ColumnTransformer(

    transformers=[

        ('num', Pipeline(steps=[

            ('imputer', SimpleImputer(strategy='mean')),  # *Imputar valores faltantes*

            ('scaler', StandardScaler())  # *Escalar características numéricas*

        ]), numeric\_features),

        ('cat', Pipeline(steps=[

            ('imputer', SimpleImputer(strategy='most\_frequent')),  # *Imputar valores faltantes*

            ('encoder', OneHotEncoder(drop='first', handle\_unknown='ignore'))  # *Codificar características categóricas*

        ]), categorical\_features)

    ]

)

# *3. Preparar las variables para el modelo*

X = df\_modelo[numeric\_features + categorical\_features]

y = df\_modelo['DESERCIÓN']  # *Suponiendo que 'DESERCIÓN' es la columna objetivo*

# *4. Dividir los datos en entrenamiento y prueba*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# *5. Crear el pipeline con los mejores parámetros encontrados*

pipeline = ImbPipeline(steps=[

    ('preprocessor', preprocessor),

    ('smote', SMOTE(random\_state=42)),  # *Aplicar SMOTE para balancear clases*

    ('classifier', RandomForestClassifier(

        n\_estimators=200,

        max\_depth=None,

        min\_samples\_split=2,

        min\_samples\_leaf=1,

        max\_features='sqrt',

        class\_weight='balanced',

        random\_state=42

    ))

])

# *6. Entrenar el modelo final*

pipeline.fit(X\_train, y\_train)

A screenshot of a computer

Description automatically generated

# *7. Evaluar el modelo final*

y\_pred = pipeline.predict(X\_test)

y\_probs = pipeline.predict\_proba(X\_test)[:, 1]

print("Matriz de confusión:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nReporte de clasificación:")

print(classification\_report(y\_test, y\_pred))

A screenshot of a computer screen

Description automatically generated

# *8. Graficar la curva ROC*

fpr, tpr, \_ = roc\_curve(y\_test, y\_probs)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc='lower right')

plt.show()

A line graph with orange and blue lines

Description automatically generated

# *9. Importancia de las características*

importances = pipeline.named\_steps['classifier'].feature\_importances\_

feature\_names = numeric\_features + list(pipeline.named\_steps['preprocessor'].transformers\_[1][1]['encoder'].get\_feature\_names\_out(categorical\_features))

importance\_df = pd.DataFrame({'Feature': feature\_names, 'Importance': importances})

importance\_df = importance\_df.sort\_values(by='Importance', ascending=False)

print("\nCaracterísticas más importantes:")

print(importance\_df.head(10))

A screen shot of a computer program

Description automatically generated

# *Visualizar las 10 características más importantes*

plt.figure(figsize=(10, 6))

sns.barplot(x='Importance', y='Feature', data=importance\_df.head(10))

plt.title('Características más importantes')

plt.show()

A graph with blue bars

Description automatically generated

# Evaluación del modelo

print("Matriz de confusión:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nReporte de clasificación:")

print(classification\_report(y\_test, y\_pred))

**Matriz de Confusión**

La matriz de confusión es:

A screenshot of numbers and symbols

Description automatically generated

* **Clase 0 (No Deserta):**
  + **Verdaderos Negativos (TN):** 53,286
  + **Falsos Positivos (FP):** 0
* **Clase 1 (Deserta):**
  + **Falsos Negativos (FN):** 0
  + **Verdaderos Positivos (TP):** 4,431

**Interpretación:**

* **TN (No Deserta correctamente clasificados):** Todos los ejemplos de la clase 0 (No Deserta) fueron correctamente clasificados como No Deserta.
* **TP (Deserta correctamente clasificados):** Todos los ejemplos de la clase 1 (Deserta) fueron correctamente clasificados como Deserta.
* **FP (Errores de clasificación de No Deserta como Deserta):** Ninguno.
* **FN (Errores de clasificación de Deserta como No Deserta):** Ninguno.

**Reporte de Clasificación**

El reporte de clasificación muestra:

A screenshot of a computer screen

Description automatically generated

* **Precisión (Precision):** La precisión es 1.00 para ambas clases. Esto significa que el modelo no ha cometido errores de clasificación. La precisión es la proporción de verdaderos positivos sobre el total de predicciones positivas.
* **Exhaustividad (Recall):** La exhaustividad es 1.00 para ambas clases. Esto significa que el modelo identificó correctamente todos los verdaderos positivos. La exhaustividad es la proporción de verdaderos positivos sobre el total de ejemplos positivos reales.
* **Puntuación F1 (F1-score):** La puntuación F1 es 1.00 para ambas clases. La puntuación F1 es la media armónica de la precisión y la exhaustividad. Una puntuación de 1.00 indica un equilibrio perfecto entre precisión y exhaustividad.
* **Exactitud (Accuracy):** La exactitud es 1.00, lo que significa que el modelo ha clasificado correctamente todos los ejemplos (sin errores).
* **Promedio Macro y Ponderado:** Ambos son 1.00, indicando que el desempeño es perfecto para ambas clases y ponderado por el tamaño de cada clase.

**Conclusiones**

* **Desempeño Perfecto:** Los valores de precisión, recall, y f1-score de 1.00 sugieren que el modelo tiene un desempeño perfecto en los datos de prueba. No cometió errores en la clasificación de ninguna de las clases.
* **Posible Sobreajuste:** Si el modelo tiene un rendimiento perfecto en los datos de entrenamiento y prueba, pero no en datos nuevos, podría estar sobreajustado. Es importante validar el modelo con datos no vistos o realizar una validación cruzada para asegurar que el desempeño generaliza bien.