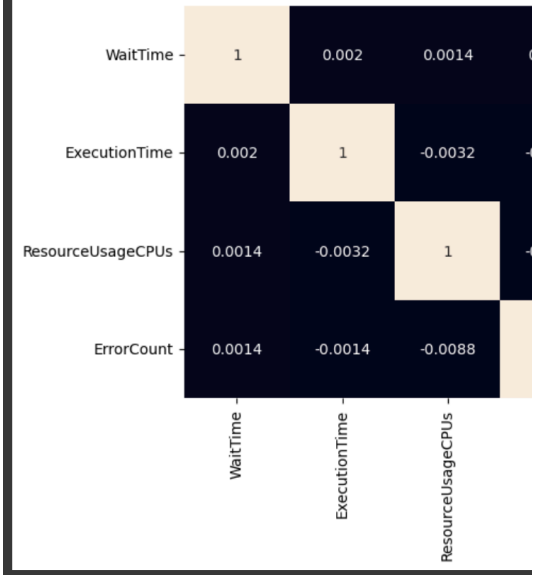
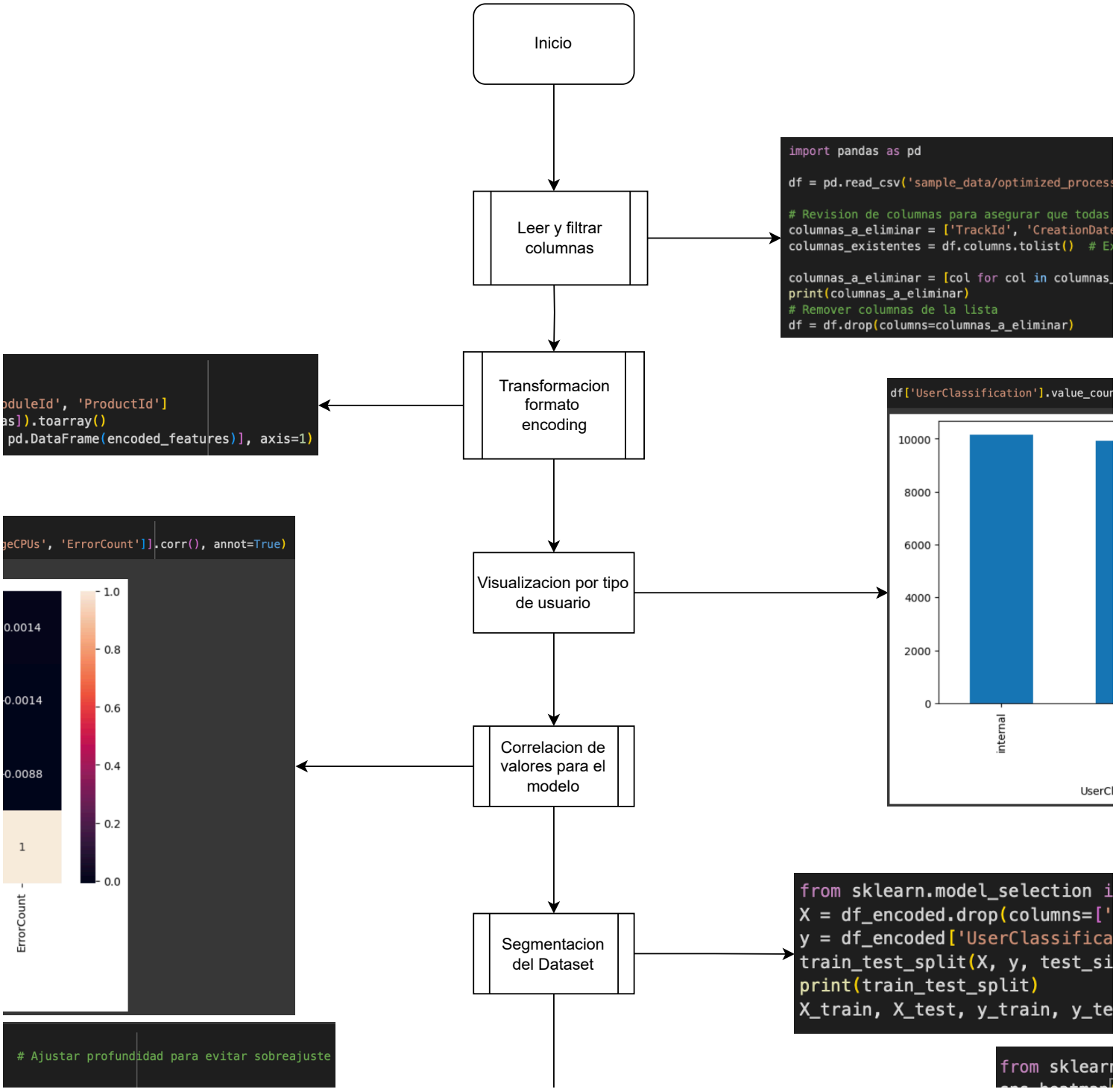
```python
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(drop='first')
categoricas = ['ProcessType', 'TransactionHealth', 'Mo
encoded_features = encoder.fit_transform(df[categorica
df_encoded = pd.concat([df.drop(columns=categoricas),
```

```python
import seaborn as sns
sns.heatmap(df[['WaitTime', 'ExecutionTime', 'ResourceUsag
```

```
<Axes: >
```

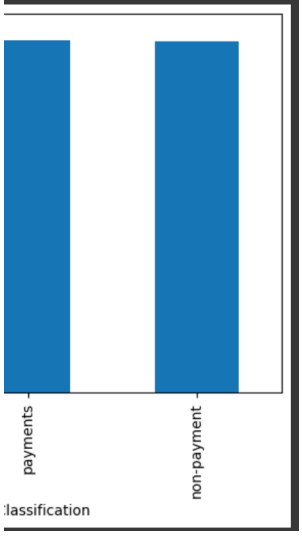



```python
from sklearn.tree import DecisionTreeClassifier
modelo = DecisionTreeClassifier(max_depth=5, random_state=42)
modelo.fit(X_train, y_train)
```


Inicio
Leer y filtrar columnas
Transformacion formato encoding
Visualizacion por tipo de usuario
Correlacion de valores para el modelo
Segmentacion del Dataset
import pandas as pd
df = pd.read_csv('sample_data/optimized_proces
# Revision de columnas para asegurar que todas
columnas_a_eliminar = ['TrackId', 'CreationDat
columnas_existentes = df.columns.tolist()
columnas_a_eliminar = [col for col in columnas_
print(columnas_a_eliminar)
# Remover columnas de la lista
df = df.drop(columns=columnas_a_eliminar)
oduleId', 'ProductId']
as]).toarray()
pd.DataFrame(encoded_features)], axis=1)
df['UserClassification'].value_cour
10000
8000
6000
4000
2000
0
internal
UserCl
geCPUs', 'ErrorCount']].corr(), annot=True)
0.0014
0.0014
0.0088
1
ErrorCount
1.0
0.8
0.6
0.4
0.2
0.0
from sklearn.model_selection i
X = df_encoded.drop(columns=['
y = df_encoded['UserClassifica
train_test_split(X, y, test_si
print(train_test_split)
X_train, X_test, y_train, y_te
Ajustar profundidad para evitar sobreajuste
from sklearn

```
ses_dataset_trans.csv')

 existan
e', 'StartDate', 'EndDate', 'UserId', 'UserName', 'UserClientName'] # Innecesarias para el arbol
xtraccion de nombres de columnas

_a_eliminar if col in columnas_existentes]  # Filter out non-existent columns
```

```
nts().plot(kind='bar')
```




```
import train_test_split
UserClassification'])
tion']
ze=0.2, random_state=42)

st = train_test_split(X, y, test_size=0.2, random_state=42) # Principio 80% y 20%
```

```
n.metrics import confusion_matrix
```

```
DecisionTreeClassifier(max_depth=5, random_state=42)
```

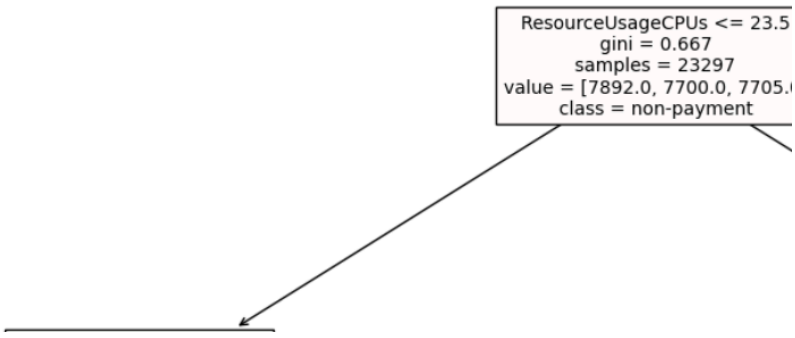
```
from sklearn.metrics import accuracy_score
y_pred = modelo.predict(X_test)
print(f'Precisión: {accuracy_score(y_test, y_pred):.2f}')
```

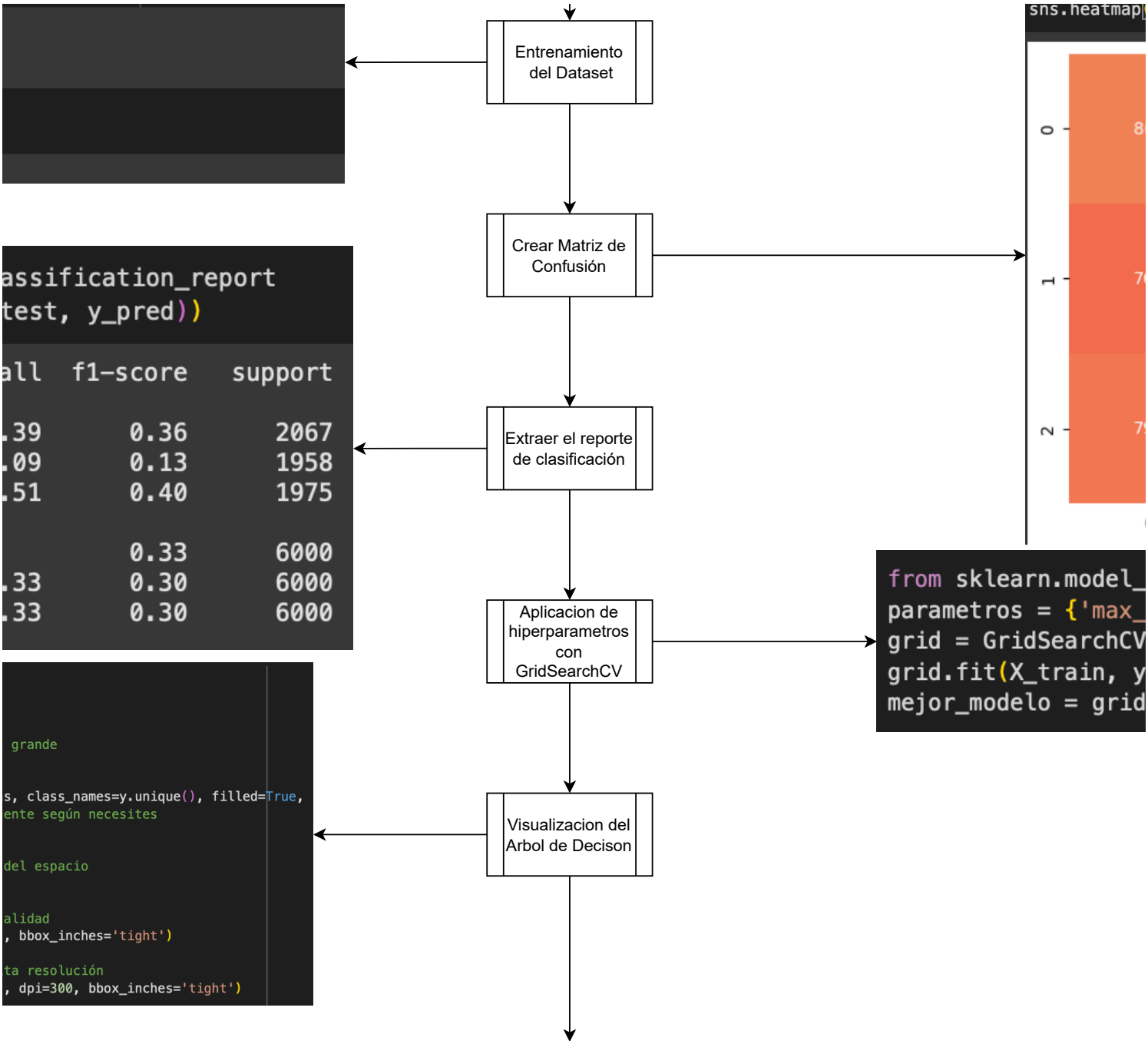
```
Precisión: 0.33
```

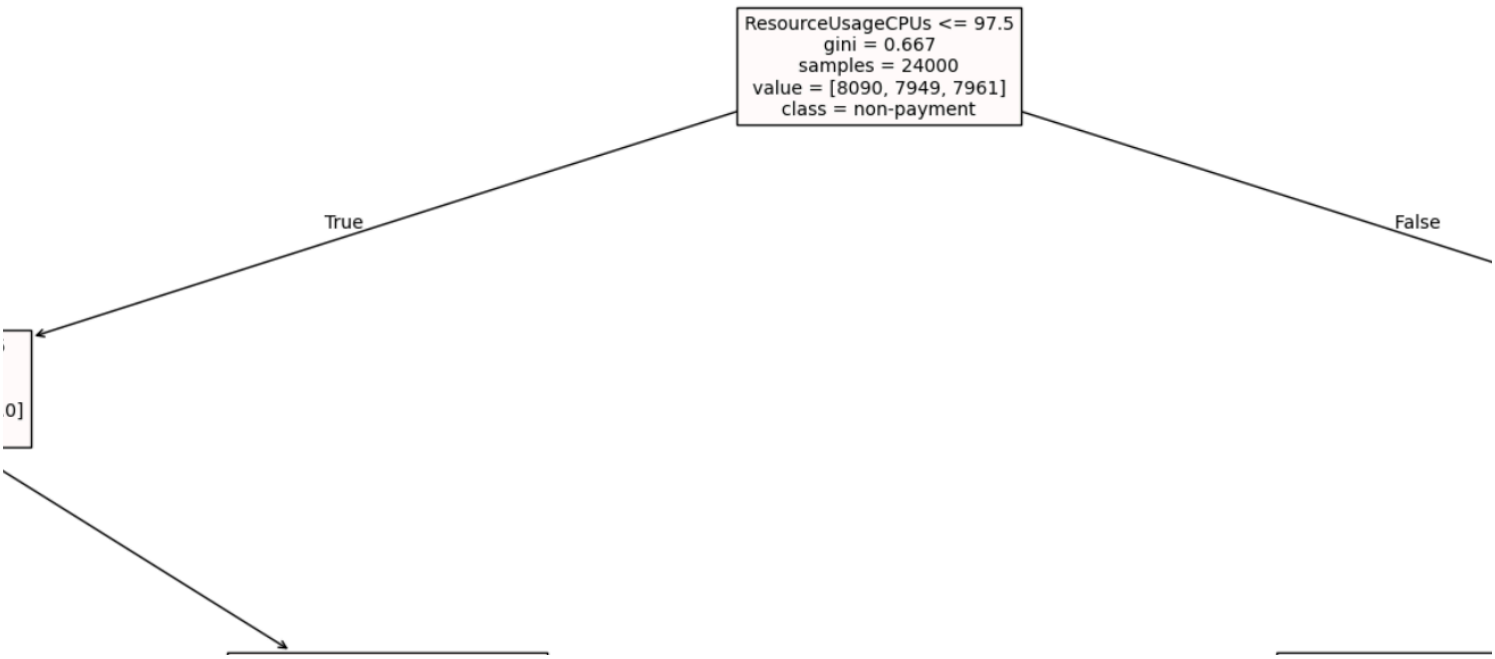
```
from sklearn.metrics import cla
print(classification_report(y_t
```

	precision	reca
internal	0.34	0.
non-payment	0.29	0.
payments	0.33	0.
accuracy		
macro avg	0.32	0.
weighted avg	0.32	0.

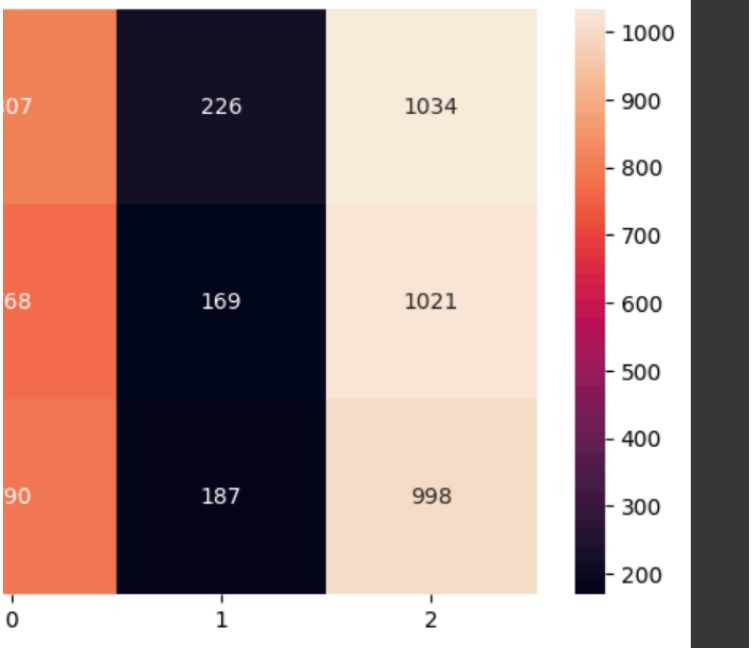
```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Aumentar el tamaño de la figura
plt.figure(figsize=(30, 15))  # Tamaño aún más

# Guardar en formato SVG para mejor zoom
plot_tree(mejor_modelo, feature_names=X.columns
          fontsize=10)  # Ajustar tamaño de fu

# Mostrar la figura
plt.tight_layout()  # Para mejor distribución
plt.show()

# Guardar como archivo SVG para zoom de alta c
plt.savefig('arbol_decision.svg', format='svg'

# Opcionalmente también guardar como PNG de al
plt.savefig('arbol_decision.png', format='png'
```



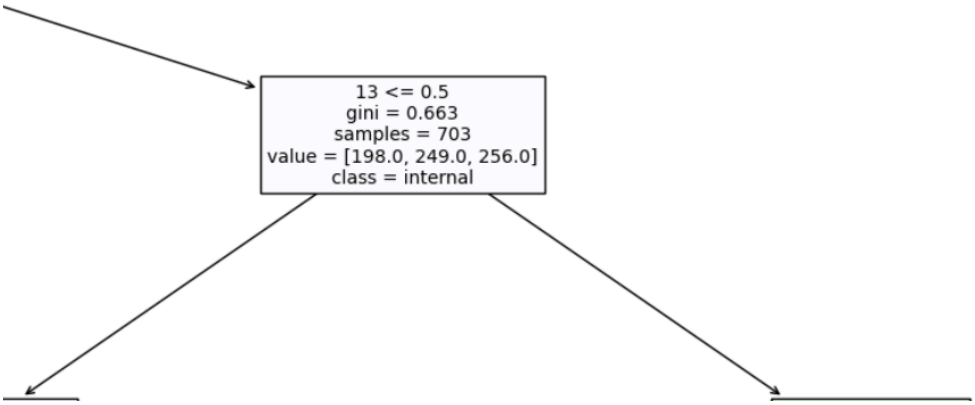

Entrenamiento del Dataset
Crear Matriz de Confusión
Extraer el reporte de clasificación
Aplicacion de hiperparametros con GridSearchCV
Visualizacion del Arbol de Decison
sns.heatmap
assification_report
test, y_pred))
all f1-score support
.39 0.36 2067
.09 0.13 1958
.51 0.40 1975
0.33 6000
.33 0.30 6000
.33 0.30 6000
from sklearn.model_
parametros = {'max_
grid = GridSearchCV
grid.fit(X_train, y
mejor_modelo = grid
grande
s, class_names=y.unique(), filled=True,
ente según necesites
del espacio
alidad
, bbox_inches='tight')
ta resolución
, dpi=300, bbox_inches='tight')

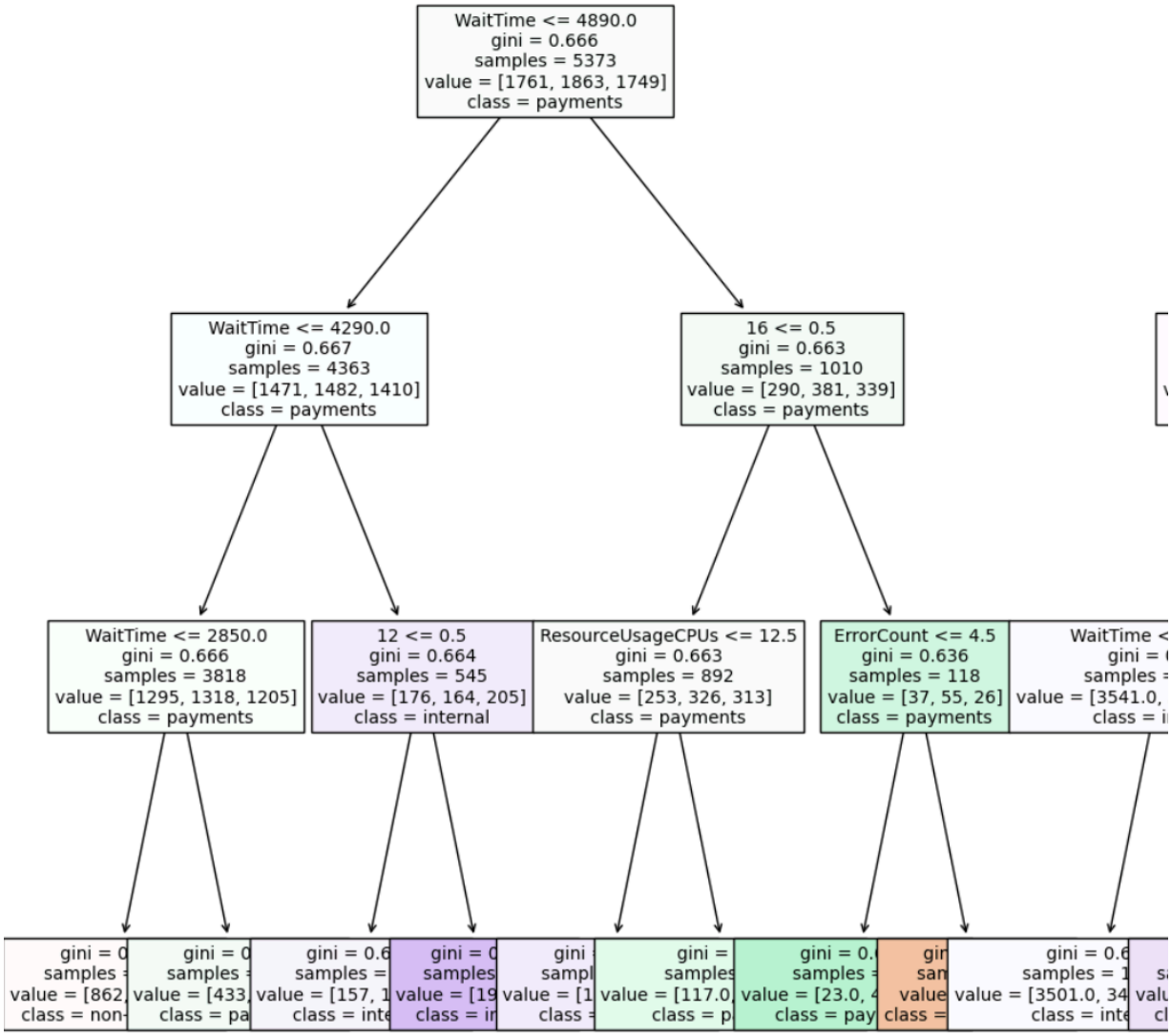

ResourceUsageCPUs <= 97.5
gini = 0.667
samples = 24000
value = [8090, 7949, 7961]
class = non-payment
True
False

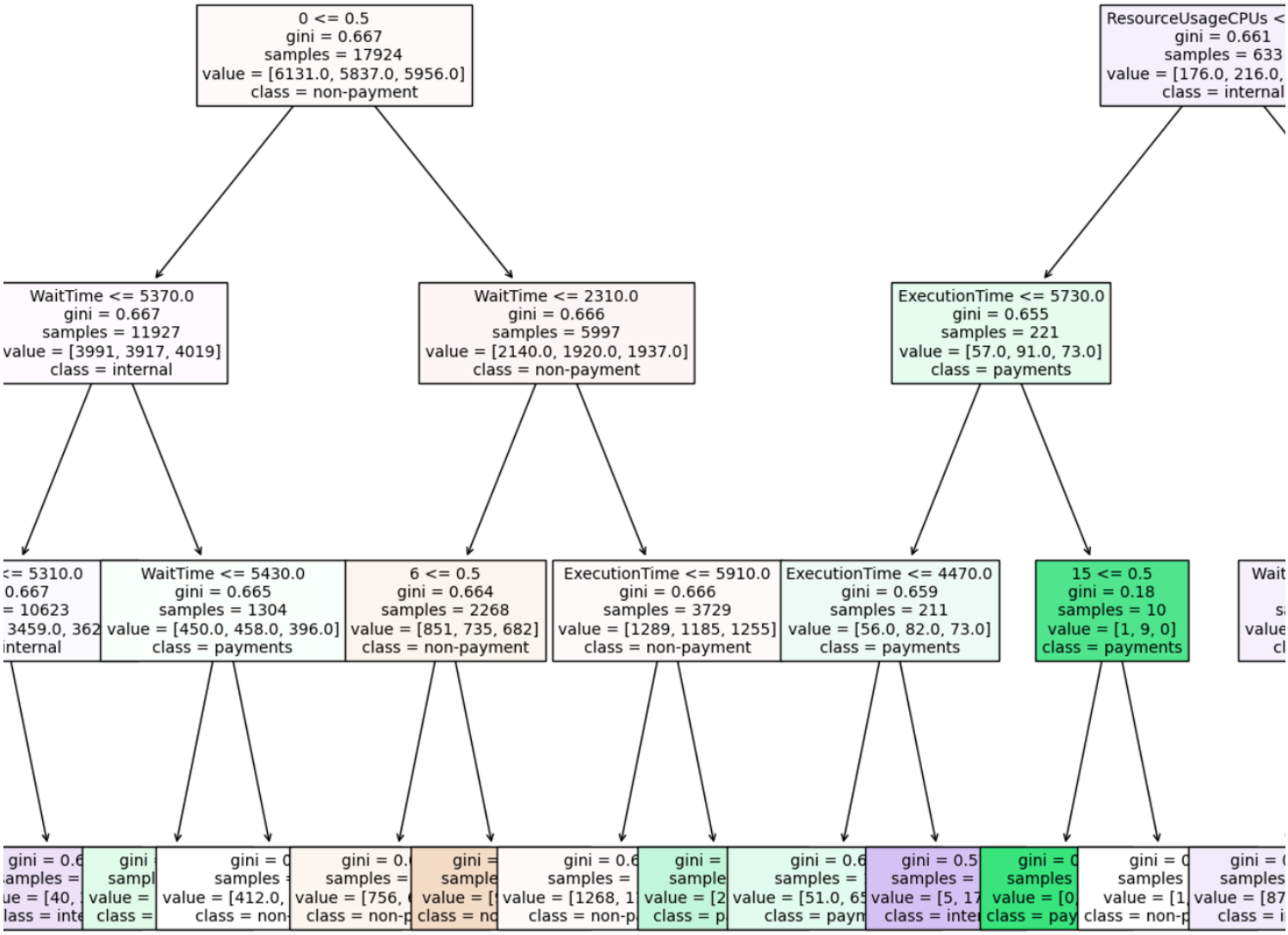
```
(confusion_matrix(y_test, y_pred), annot=True, fmt='d')
```

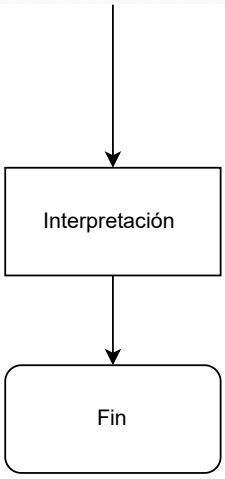


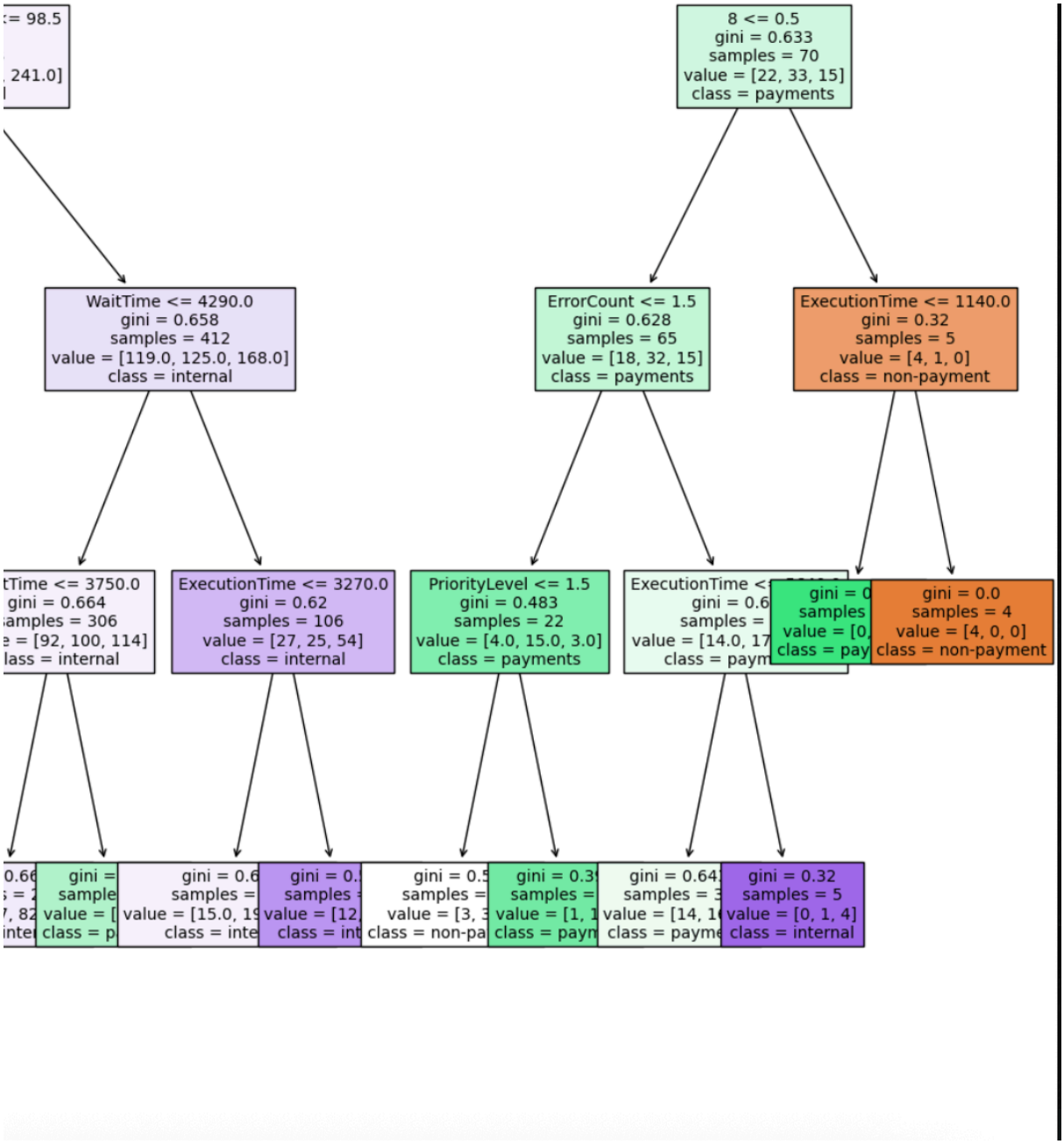

```
selection import GridSearchCV
depth': [3, 5, 7], 'min_samples_split': [2, 5, 10]}
(DecisionTreeClassifier(), parametros, cv=5)
_train)
.best_estimator_
```

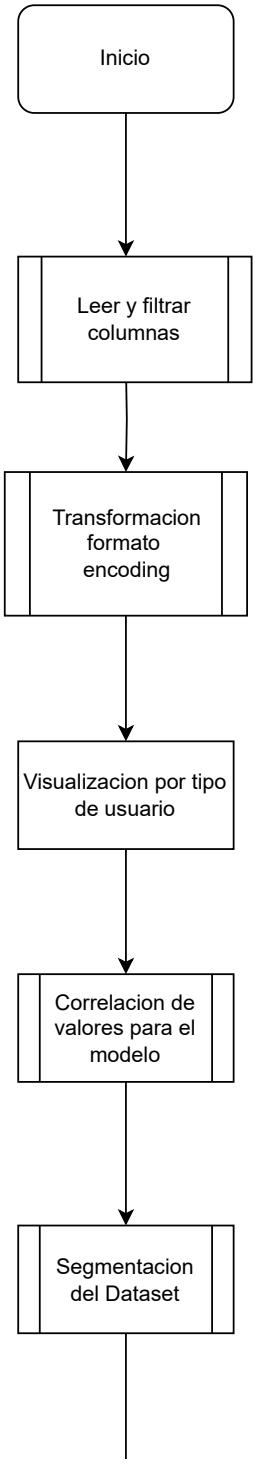


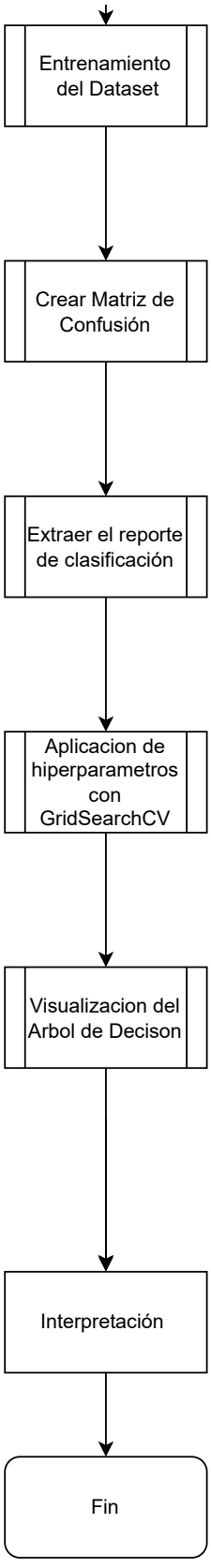

WaitTime <= 4890.0
gini = 0.666
samples = 5373
value = [1761, 1863, 1749]
class = payments
WaitTime <= 4290.0
gini = 0.667
samples = 4363
value = [1471, 1482, 1410]
class = payments
16 <= 0.5
gini = 0.663
samples = 1010
value = [290, 381, 339]
class = payments
WaitTime <= 2850.0
gini = 0.666
samples = 3818
value = [1295, 1318, 1205]
class = payments
12 <= 0.5
gini = 0.664
samples = 545
value = [176, 164, 205]
class = internal
ResourceUsageCPUs <= 12.5
gini = 0.663
samples = 892
value = [253, 326, 313]
class = payments
ErrorCount <= 4.5
gini = 0.636
samples = 118
value = [37, 55, 26]
class = payments


0 <= 0.5
gini = 0.667
samples = 17924
value = [6131.0, 5837.0, 5956.0]
class = non-payment
ResourceUsageCPUs <
gini = 0.661
samples = 633
value = [176.0, 216.0,
class = internal
WaitTime <= 5370.0
gini = 0.667
samples = 11927
value = [3991, 3917, 4019]
class = internal
WaitTime <= 2310.0
gini = 0.666
samples = 5997
value = [2140.0, 1920.0, 1937.0]
class = non-payment
ExecutionTime <= 5730.0
gini = 0.655
samples = 221
value = [57.0, 91.0, 73.0]
class = payments
<= 5310.0
0.667
= 10623
3459.0, 362
internal
WaitTime <= 5430.0
gini = 0.665
samples = 1304
value = [450.0, 458.0, 396.0]
class = payments
6 <= 0.5
gini = 0.664
samples = 2268
value = [851, 735, 682]
class = non-payment
ExecutionTime <= 5910.0
gini = 0.666
samples = 3729
value = [1289, 1185, 1255]
class = non-payment
ExecutionTime <= 4470.0
gini = 0.659
samples = 211
value = [56.0, 82.0, 73.0]
class = payments
15 <= 0.5
gini = 0.18
samples = 10
value = [1, 9, 0]
class = payments
Wait
value
cl


Interpretación
Fin


= 98.5
241.0]
8 <= 0.5
gini = 0.633
samples = 70
value = [22, 33, 15]
class = payments
WaitTime <= 4290.0
gini = 0.658
samples = 412
value = [119.0, 125.0, 168.0]
class = internal
ErrorCount <= 1.5
gini = 0.628
samples = 65
value = [18, 32, 15]
class = payments
ExecutionTime <= 1140.0
gini = 0.32
samples = 5
value = [4, 1, 0]
class = non-payment
tTime <= 3750.0
gini = 0.664
amples = 306
e = [92, 100, 114]
lass = internal
ExecutionTime <= 3270.0
gini = 0.62
samples = 106
value = [27, 25, 54]
class = internal
PriorityLevel <= 1.5
gini = 0.483
samples = 22
value = [4.0, 15.0, 3.0]
class = payments
gini = 0.0
samples = 4
value = [4, 0, 0]
class = non-payment
gini = 0.32
samples = 5
value = [0, 1, 4]
class = internal


Inicio
Leer y filtrar columnas
Transformacion formato encoding
Visualizacion por tipo de usuario
Correlacion de valores para el modelo
Segmentacion del Dataset


Entrenamiento del Dataset
Crear Matriz de Confusión
Extraer el reporte de clasificación
Aplicacion de hiperparametros con GridSearchCV
Visualizacion del Arbol de Decison
Interpretación
Fin