In [ ]:
```
!pip install pyparsing
```

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt

import numpy as np

from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz

import os
from graphviz import Source

#     :::PRUEBAS:::
from pydotplus import graph_from_dot_data
from IPython.display import Image
```

Lectura Datos

In [112]:
```
data = pd.read_csv("TEST 0.csv",delimiter=";")
data.head()
```

Out[112]:

	estu_consecutivo.1	estu_exterior	periodo	estu_tieneetnia	estu_tomo_cursopreparacion	estu_cursodocentesies	estu_cursoiesapoyoexte
0	SB11201320218705	NO	20183	No	NaN	NaN	I
1	SB11201220483104	NO	20183	No	NaN	NaN	I
2	SB11201320115727	NaN	20152	NaN	No	No tomó Curso	No tomó Cu
3	SB11201210035597	NO	20173	No	NaN	NaN	I
4	SB11201320132366	NO	20173	No	NaN	NaN	I

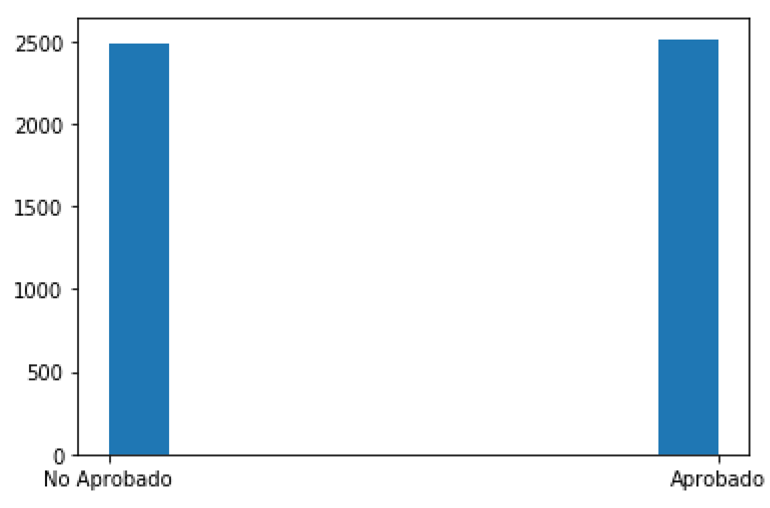
5 rows × 78 columns

In [113]:
```
#Dastos del DataFrame
data.shape
```

Out[113]:
```
(5000, 78)
```

In [114]:
```
data.exito.replace({0:"No Aprobado",1:"Aprobado"},inplace=True)
```

In [115]:
```
#Histograma de Casos de Exito
plt.hist(data.exito)
plt.show()
```



In [116]:
```
data.exito.unique()
```

Out[116]:
```
array(['No Aprobado', 'Aprobado'], dtype=object)
```

In [117]:
```
colnames = data.columns.values.tolist()
colnames
```

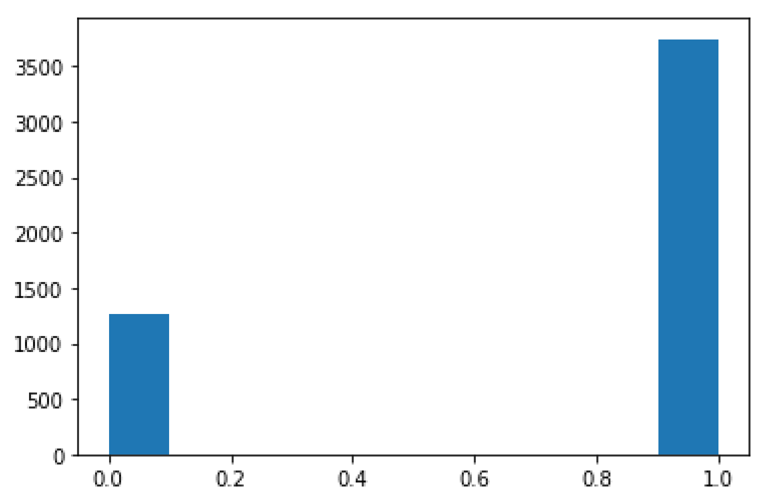
Out[117]:
```
['estu_consecutivo.1',
 'estu_exterior',
 'periodo',
 'estu_tieneetnia',
 'estu_tomo_cursopreparacion',
 'estu_cursodocentesies',
 'estu_cursoiesapoyoexterno',
 'estu_cursoiesexterna',
 'estu_simulacrotipoicfes',
 'estu_actividadrefuerzoareas',
 'estu_actividadrefuerzogeneric',
 'fami_trabajolaborpadre',
 'fami_trabajolabormadre',
 'fami_numlibros',
 'estu_inst_cod_departamento',
 'estu_tipodocumento.1',
 'estu_nacionalidad.1',
 'estu_genero.1',
 'estu_fechanacimiento.1',
 'periodo.1',
 'estu_estudiante.1',
 'estu_pais_reside.1',
 'estu_depto_reside.1',
 'estu_cod_reside_depto.1',
 'estu_mcpio_reside.1',
 'estu_cod_reside_mcpio.1',
 'estu_areareside',
 'estu_valorpensioncolegio',
 'fami_educacionpadre.1',
 'fami_educacionmadre.1',
 'fami_ocupacionpadre.1',
 'fami_ocupacionmadre.1',
 'fami_estratovivienda.1',
 'fami_nivelsisben',
 'fami_pisoshogar',
 'fami_tieneinternet.1',
 'fami_tienecomputador.1',
 'fami_tienemicroondas',
 'fami_tienehorno',
 'fami_tieneautomovil.1',
 'fami_tienedvd',
 'fami_tiene_nevera.1',
 'fami_tiene_celular.1',
 'fami_telefono.1',
 'fami_ingresofmiliarmensual',
 'estu_trabajaactualmente',
 'estu_antecedentes',
 'estu_expectativas',
 'cole_codigo_icfes',
 'cole_cod_dane_establecimiento',
 'cole_nombre_establecimiento',
 'cole_genero',
 'cole_naturaleza',
 'cole_calendario',
 'cole_bilingue',
 'cole_caracter',
 'cole_cod_dane_sede',
 'cole_nombre_sede',
 'cole_sede_principal',
 'cole_area_ubicacion',
 'cole_jornada',
 'cole_cod_mcpio_ubicacion',
 'cole_mcpio_ubicacion',
 'cole_cod_depto_ubicacion',
 'cole_depto_ubicacion',
 'punt_lenguaje',
 'punt_matematicas',
 'punt_biologia',
 'punt_quimica',
 'punt_fisica',
 'punt_ciencias_sociales',
 'punt_filosofia',
 'punt_ingles',
 'desemp_ingles',
 'profundiza',
 'puntaje_prof',
 'desemp_prof',
 'exito']
```

In [154]:
```
#Variable Objetivo -> Exito
#Predictoras -> puntaje
target = colnames[-1]
predictors = colnames[65:73]
predictors
```

Out[154]:
```
['punt_lenguaje',
 'punt_matematicas',
 'punt_biologia',
 'punt_quimica',
 'punt_fisica',
 'punt_ciencias_sociales',
 'punt_filosofia',
 'punt_ingles']
```

In [144]:
```
#75% -> Entrenar
#25% -> Validar
data["is_train"] = np.random.uniform(0,1, len(data))<=0.75
```

In [146]:
```
#Validar -> 1250
#Entrenar -> 3750
plt.hist(data.is_train.astype(np.int))
plt.show()
```



In [129]:
```
train,test = data[data["is_train"]==True],data[data["is_train"]==False]
```

In [148]:
```
#min_samples_split -> Minimo de muestras para formar
    #el árbol

tree = DecisionTreeClassifier(criterion="gini",
                              max_depth=4,
                               min_samples_split=20,
                               random_state=1)
#Modelado
tree.fit(train[predictors],train[target])
```

Out[148]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=20,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=1, splitter='best')
```

In [149]:
```
#predecir del 25%
preds = tree.predict(test[predictors])
```

In [152]:
```
# Datos actual del datset vs prediccion del modelo
pd.crosstab(test[target],preds,rownames=["Actual"],
           colnames=["Predictions"])
```

Out[152]:

Predictions	Aprobado	No Aprobado
Actual		
Aprobado	466	170
No Aprobado	140	525

Visualización del Árbol de Decisión

In [153]:
```
#
os.environ['PATH'] = os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+r"\Library\bin\graphviz"

dot_data = export_graphviz(tree,out_file=None,
                          filled=True,rounded=True,
                          class_names=["No Aprobado",
                                       "Aprobado"],
                          feature_names=predictors)
graph = graph_from_dot_data(dot_data)
graph.write_png("tree.png")
Image(graph.create_png())
```

Out[153]: