Campus Monterrey

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución. (Portafolio Implementación)

Juan Pablo Castañeda Serrano A01752030

Dataset

El dataset utilizado fue uno parecido al reto, llamado “Spaceship Titanic”(https://www.kaggle.com/competitions/spaceship-titanic) este fue escogido debido a mi familiarización con los modelos de clasificación y la estructura de sus datos.
Este dataset incluye diferentes variables, sin embargo para el entrenamiento del modelo se realizó solo con las siguientes:
[CryoSleep,Age,VIP,RoomService,FoodCourt,ShoppingMall,Spa,VRDeck]
Esto ya que son las variables con mayor porcentaje de correlación y tienen valores numéricos con pocos o nulos datos NA. Todos los demás fueron eliminados, la excepción fueron las variables de las ids de los tripulantes y su resultado final, las cuales no fueron consideradas en el entrenamiento pero fueron usadas para concatenar los valores finales con su respectivo id de tripulante

Modelo

El problema que tenemos con este dataset es similar al del reto, el poder predecir si un pasajero abordará la nave o si no. En primera instancia, podemos observar que la predicción está puesta como “False” y “True” en lugar de binarios, decidí convertir el false y el true a binario y luego volver a transformarlo a string en el momento de la concatenación. Como este problema es un problema de clasificación, sentí que el Desition Tree Clasifier. Un clasificador de árbol de decisión es un modelo de aprendizaje automático supervisado que se utiliza tanto para tareas de clasificación como de regresión. Funciona dividiendo datos de forma recursiva en subconjuntos hasta que alcanza algún criterio de terminación. El conjunto de datos sobre la nave espacial tiene una combinación de características categóricas y numéricas, Decision Trees puede manejar ambas sin mucho problema y esa fue otra razón para utilizar estas.

Entrenamiento y predicción

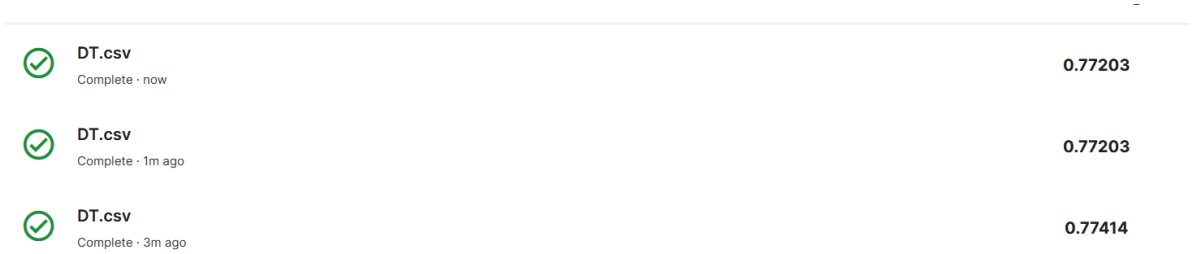
Las variables utilizadas en el modelo para crear diferentes predicciones fueron el cambio de los parámetros para la ecuación de predicción, las que se usaron fueron: “gini”, “entropy” y “log_loss”. De esta manera se hicieron 3 predicciones diferentes, sin embargo los resultados fueron equivalentes, siendo gini el que mejores resultados proveyó. Estos son los resultados:

Gini con 77.4% de accuracy:

Log-Loss con 77.2%

Entropy con 77.2%

Es interesante remarcar que Entropy y Log Loss produjeron virtualmente los mismos resultados



Partes relevantes del código

Presente en el archivo “predict.py” y “Clean.py”

Limpieza de los datos:

```python
import pandas as pd

def clean(name):
    df_train = pd.read_csv(name, encoding='unicode_escape', engine='python')

    df_train.drop(['HomePlanet','Cabin', 'Destination', 'Name'], axis=1, inplace=True)

    #df_train.dropna(inplace=True)

    mappingbin = {'False': 0, 'True': 1}
    df_train.replace({'CryoSleep': mappingbin, 'VIP': mappingbin}, inplace=True)

    return df_train

if __name__ == '__main__':
    clean("train.csv")
```

Abrir y entrenar:

```python
df = clean('train.csv')
dfTest = clean('test.csv')
ids = dfTest.PassengerId
x = df.drop(['PassengerId','Transported'],axis=1)
x2 = dfTest.drop(['PassengerId'],axis=1)
y_train= df['Transported']

scaler=MinMaxScaler()
scaler.fit(x)

x_train_transformada=scaler.transform(x)
x_test_transformada=scaler.transform(x2)
```

Modelo:

```
def DTree():
    model_t=DecisionTreeClassifier(random_state=10,criterion="entropy")

    model_t.fit(x_train_transformada,y_train)

    y_hat_t=model_t.predict(x_test_transformada)
    return y_hat_t
```

Graficas y concatenación

```
def graficar():
    model_t = DecisionTreeClassifier(random_state=10,criterion="entropy")
    model_t.fit(x_train_transformada,y_train)
    myTreeData = sklearn.tree.export_graphviz(model_t)
    graphData = graphviz.Source(myTreeData)
    graphData.render("data.gv")




def concatenar(ids,emission,name):
    final=pd.DataFrame()
    final['PassengerId'] = ids['PassengerId']
    final['Transported']=pd.Series(emission)
    final['Transported']=final['Transported']
    final.to_csv(name,index=False)
    print("checa tu csv con nombre: "+name)
    return final
```

Llamada final

```
if __name__=='__main__':
    #DT = DTree()
    #concatenar(dfTest,DT,'DT.csv')
    graficar()
```