

P2. Clasificación

Para este proyecto, se desarrollan y comparan distintos modelos de clasificación para predecir si un pasajero fue transportado en la nave "Spaceship Titanic", basándose en la información proveída por Kaggle. Se aplican diferentes técnicas de Machine Learning, incluyendo regresión logística multinomial, análisis discriminante lineal (LDA), árboles de decisión y métodos de ensamble como bagging, random forest y boosting.

El flujo de trabajo sigue estos pasos:

- Carga y exploración de datos: Se analizan los datos proporcionados, identificando valores nulos y características relevantes.
- Entrenamiento de modelos de clasificación: Se implementan y evalúan diversos modelos supervisados para identificar cuál ofrece el mejor rendimiento.
- Validación y comparación de modelos: Se utilizan técnicas de validación cruzada para medir la precisión de cada modelo y determinar su efectividad.
- Predicción y envío de resultados: Se genera el archivo de predicciones finales y se somete a la evaluación de Kaggle.

```
In [74]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

1. Se cargan los datos de entrenamiento y prueba, eliminando columnas irrelevantes como PassengerId, Name y Cabin. Luego, se codifican las variables categóricas con LabelEncoder() y se normalizan los datos numéricos con StandardScaler() para asegurar que todos los modelos trabajen con valores en la misma escala.

Igualmente, se separan los datos en X_train y y_train para el entrenamiento, y en X_val y y_val para la validación. Se utiliza un test_size=0.2, lo que significa que el 20% de los datos se reserva para validar los modelos antes de hacer predicciones finales.

```
In [66]: # Cargar datos
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')

# Exploración de datos
print(df_train.head())
print(df_train.info())
print(df_train.isnull().sum())
```

```
# Preprocesamiento de datos
# Rellenar valores nulos y convertir variables categóricas
# Encode categorical variables
for col in ['HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'VIP']:
    df_train[col].fillna(df_train[col].mode()[0], inplace=True)
    df_test[col].fillna(df_test[col].mode()[0], inplace=True)

le = LabelEncoder()
combined_data = pd.concat([df_train[col], df_test[col]], axis=0) # Combine train
le.fit(combined_data) # Fit on combined data

df_train[col] = le.transform(df_train[col]) # Transform train
df_test[col] = le.transform(df_test[col]) # Transform test
```

```

PassengerId HomePlanet CryoSleep Cabin Destination Age VIP \
0    0001_01    Europa    False  B/O/P TRAPPIST-1e 39.0 False
1    0002_01     Earth    False  F/O/S TRAPPIST-1e 24.0 False
2    0003_01    Europa    False  A/O/S TRAPPIST-1e 58.0 True
3    0003_02    Europa    False  A/O/S TRAPPIST-1e 33.0 False
4    0004_01     Earth    False  F/I/S TRAPPIST-1e 16.0 False

RoomService FoodCourt ShoppingMall Spa VRDeck           Name \
0        0.0      0.0       0.0   0.0      0.0 Maham Ofracculy
1     109.0      9.0     25.0  549.0     44.0 Juanna Vines
2      43.0    3576.0       0.0 6715.0     49.0 Altark Susent
3        0.0    1283.0     371.0 3329.0    193.0 Solam Susent
4     303.0      70.0    151.0  565.0      2.0 Willy Santantines

Transported
0    False
1    True
2   False
3   False
4    True
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column      Non-Null Count Dtype  
--- 
 0   PassengerId  8693 non-null  object 
 1   HomePlanet   8492 non-null  object 
 2   CryoSleep    8476 non-null  object 
 3   Cabin        8494 non-null  object 
 4   Destination  8511 non-null  object 
 5   Age          8514 non-null  float64
 6   VIP          8490 non-null  object 
 7   RoomService  8512 non-null  float64
 8   FoodCourt    8510 non-null  float64
 9   ShoppingMall 8485 non-null  float64
 10  Spa          8510 non-null  float64
 11  VRDeck       8505 non-null  float64
 12  Name         8493 non-null  object 
 13  Transported  8693 non-null  bool   
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
None
PassengerId      0
HomePlanet      201
CryoSleep       217
Cabin          199
Destination     182
Age            179
VIP             203
RoomService     181
FoodCourt       183
ShoppingMall    208
Spa             183
VRDeck          188
Name            200

```

```
Transported      0  
dtype: int64
```

```
In [67]: # Convertir 'Transported' en binario  
df_train['Transported'] = df_train['Transported'].astype(int)  
  
# Selección de características y target  
features = ['HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'VIP']  
X = df_train[features]  
y = df_train['Transported']  
X_test = df_test[features]  
  
# Normalización de los datos  
scaler = StandardScaler()  
X = scaler.fit_transform(X)  
X_test = scaler.transform(X_test)  
  
# División de datos  
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2)
```

2. Se entrena un modelo de regresión logística multinomial y se evalúa su desempeño utilizando validación cruzada con 5 folds. La métrica utilizada es la accuracy, y el resultado final es el promedio de las 5 evaluaciones (np.mean(cross_val_score(...))).

```
In [68]: BestModel = 0  
BestScore = 0  
  
def getBestModel(model, score, X_train, y_train):  
    global BestModel, BestScore  
  
    if(BestModel):  
        BestScore = np.mean(cross_val_score(BestModel, X_train, y_train, cv=5, scoring='accuracy'))  
  
    if(BestScore < score):  
        model.fit(X_train, y_train)  
        BestModel = model  
  
# Modelos de clasificación  
def evaluate_model(model, X_train, y_train):  
    score = np.mean(cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy'))  
  
    # Esto guardará el mejor modelo que el modelo guardará al final.  
    getBestModel(model, score, X_train, y_train)  
  
    print(f'Accuracy: {score:.4f}')  
    return model  
  
  
print("Regresión Logística:")  
log_reg = evaluate_model(LogisticRegression(), X_train, y_train)
```

Regresión Logística:
Accuracy: 0.7193

3. Se implementa un modelo de LDA, que es útil para problemas de clasificación con datos gaussianos. Se evalúa su rendimiento de la misma forma que la regresión logística, utilizando validación cruzada con 5 folds.

```
In [69]: print("LDA:")
lda = evaluate_model(LinearDiscriminantAnalysis(), X_train, y_train)
```

```
LDA:
Accuracy: 0.7207
```

4. Se entrena un modelo basado en árboles de decisión y se evalúa utilizando la misma metodología de validación cruzada. Se compara su desempeño con los modelos anteriores para ver si es más efectivo en la clasificación de los datos.

```
In [70]: print("Árbol de Decisión:")
dt = evaluate_model(DecisionTreeClassifier(), X_train, y_train)
```

```
Árbol de Decisión:
Accuracy: 0.6415
```

5. Se entrena un modelo de Random Forest, el cual utiliza una técnica de bagging al combinar múltiples árboles de decisión con el objetivo de mejorar la precisión. Se evalúa con validación cruzada y se compara su desempeño con los otros modelos. Igualmente, se repite el mismo proceso pero con Gradient Boosting.

```
In [71]: print("Random Forest:")
rf = evaluate_model(RandomForestClassifier(), X_train, y_train)

print("Gradient Boosting:")
gb = evaluate_model(GradientBoostingClassifier(), X_train, y_train)
```

```
Random Forest:
Accuracy: 0.6424
Gradient Boosting:
Accuracy: 0.7140
```

6. Se utiliza el mejor modelo para predecir la variable objetivo en los datos de prueba (test.csv). Se genera el **archivo de salida** en el formato requerido por Kaggle y se obtiene la métrica de accuracy.

```
In [72]: # Predicción en datos de prueba
final_model = BestModel
y_test_pred = final_model.predict(X_test)

# Generar archivo para Kaggle
submission = pd.DataFrame({'PassengerId': df_test['PassengerId'], 'Transported': y_
submission['Transported'] = submission['Transported'].astype(bool)
submission.to_csv('submission.csv', index=False)

print("Archivo submission.csv generado correctamente.")
```

Archivo submission.csv generado correctamente.

De acuerdo con los resultados obtenidos del accuracy de cada modelo, se puede determinar que LDA fue el mejor modelo con un valor de 0.7207. Es posible que el problema sea linealmente separable y no se necesite un modelo más complejo.

```
In [73]: # El mejor modelo es:  
print("El mejor modelo del conjunto fue: ", BestModel)
```

```
El mejor modelo del conjunto fue: LinearDiscriminantAnalysis()
```

Código de honor: Doy mi palabra de que he realizado esta actividad con integridad académica.