Inteligencia Artificial

Proyecto Final

Integrantes:
María Lucía Velasquez Peña
Sebastián Mora Geovo

Predicción de popularidad de canciones en Spotify

Mayo de 2023

Introducción

En este proyecto se usará el dataset *Music Recommendation System using Spotify* disponible en Kaggle. Se usará la información de este conjunto de datos para construir un modelo de aprendizaje automático capaz de predecir la popularidad de una canción lanzada en Spotify. Inicialmente se usará la variable "popularity" como entrada y las demás variables como características. En términos de la limpieza y el preprocesamiento de datos, se eliminarán las columnas que no son relevantes para el análisis, eliminar las filas que tengan valores faltantes, normalizar o estandarizar las variables numéricas para asegurar que todas tengan un rango similar de valores, y finalmente se entrenarían diferentes modelos de aprendizaje automático como regresión lineal o SVM para lograr la predicción en cuestión. Con base en esto, se podría evaluar el desempeño de cada modelo usando métricas como el coeficiente de determinación o el error cuadrático medio.

Ya que el *dataset* es un originario de Spotify aquí está el link para ver una mejor explicación de este:

https://developer.spotify.com/documentation/web-api/reference/get-audio-features.

Objetivos

- Realizar una limpieza de columna, datos y valores que estén incompletos y/o que no sean de relevancia, los cuales nos dificulten la realización del proyecto, además de una normalización de los datos relevantes para un correcto manejo del dataset.
- Realizar una comparación de diferentes modelos de aprendizaje automático, como regresión lineal, SVM, para predecir la popularidad de las canciones en el dataset. Se deberán evaluar y comparar los resultados obtenidos por cada modelo mediante el cálculo del coeficiente de determinación (R^2) y otras métricas relevantes.

Resultados

- **Limpieza del Dataset:** Para poder realizar la limpieza del dataset lo primero que se hizo fue eliminar las filas que estaban en blanco y/o con caracteres que no fueran números y filas que no nos sirvieran para el proyecto. Luego, el proyecto tenía un problema con la columna de *tempo* y *loudness* debido a que la representación del número era incorrecta y no se sabía con claridad cuántos datos eran los que estaban, por lo que se realizó una limpieza. A través de esta se encontró que fueron entre 80 a 100 celdas, lo cual no se consideró como algo crítico, pues el *dataset* tiene 25.000 celdas en total.
- **Código:** A continuación, se presenta el código realizado, debidamente documentado y posteriormente se presenta la explicación.

```
import pandas as pd
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error

# Carga el dataset
data = pd.read_csv("SpotifySongP.csv", error_bad_lines=False, delimiter=',')
df_clean = data.dropna(how='any')

# Eliminar filas con valores no numéricos en 'popularity'
df_clean = df_clean[~df_clean['popularity'].isin(['ScreaMER'])]

# Eliminar filas con valores faltantes en 'popularity'
df_clean = df_clean.dropna(subset=['popularity'])
df_clean = df_clean.drop(27886)

# Imputar valores faltantes en 'popularity'
imputer = SimpleImputer(strategy='most_frequent')
df_clean['popularity'] = imputer.fit_transform(df_clean[['popularity']])

# Separar características (X) y variable objetivo (y)
X = df_clean[['danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms']]
y = df_clean['popularity']

# Definir modelos y parámetros para GridSearchCV
models = [
    ('Linear Regression', LinearRegression(), {}),
    ('Ridge', Ridge(), {'alpha': [0.1, 1.0, 10.0]}),
    ('Lasso', Lasso(), {'alpha': [0.1, 1.0, 10.0]}),
    ('Random Forest', RandomForestRegressor(), {'n_estimators': [50, 100, 200]})
]
```

```
# Definir modelos y parámetros para GridSearchCV
models = [
    ('Linear Regression', LinearRegression(), {}),
    ('Ridge', Ridge(), {'alpha': [0.1, 1.0, 10.0]}),
    ('Lasso', Lasso(), {'alpha': [0.1, 1.0, 10.0]}),
    ('Random Forest', RandomForestRegressor(), {'n_estimators': [50, 100, 200]})
]

# Realizar validación cruzada con GridSearchCV
best_models = []
best_scores = []
best_params = []

for name, model, params in models:
    grid_search = GridSearchCV(model, params, cv=5, scoring='neg_mean_squared_error')
    grid_search.fit(X, y)
    best_models.append(name)
    best_scores.append(grid_search.best_score_)
    best_params.append(grid_search.best_params_)

# Imprimir los mejores modelos, puntajes y parámetros
for name, score, params in zip(best_models, best_scores, best_params):
    print(f"Modelo: {name}")
    print(f"Neg-MSE: {score}")
    print(f"RMSE: {abs(score) ** 0.5}")
    print(f"Mejores parámetros: {params}")
    print()
```

- **Valores obtenidos**

```
Modelo: Linear Regression
Puntaje: -0.29596330404741517
Mejores parámetros: {}

Modelo: Ridge
Puntaje: -0.2958335541510818
Mejores parámetros: {'alpha': 10.0}

Modelo: Lasso
Puntaje: -0.29422777240872444
Mejores parámetros: {'alpha': 0.1}

Modelo: Random Forest
Puntaje: -0.2432315608770897
Mejores parámetros: {'n_estimators': 200}
```

```
 data = pd.read_csv("SpotifySongP.csv", error_bad_lines=False, delimiter=',')
Modelo: Linear Regression
Neg-MSE: -409.23225691079836
RMSE: 20.229489783748832
Mejores parámetros: {}

Modelo: Ridge
Neg-MSE: -409.2145178262232
RMSE: 20.229051332828814
Mejores parámetros: {'alpha': 10.0}

Modelo: Lasso
Neg-MSE: -410.9271983131269
RMSE: 20.271339332000906
Mejores parámetros: {'alpha': 0.1}

Modelo: Random Forest
Neg-MSE: -383.54391206106504
RMSE: 19.584277164630432
Mejores parámetros: {'n_estimators': 200}
```

- **Explicación de regresiones**

  Como se puede ver en el código, se utilizan cuatro modelos diferentes de regresión. Estos evalúan y comparan el rendimiento en la predicción de la popularidad de las canciones en función de las características proporcionadas. En primera instancia, la regresión lineal se usa como uno de los modelos de referencia para comparar su rendimiento, con respecto a otros modelos más complejos. Adicionalmente, la regresión de Ridge se usa para introducir una penalización de los coeficientes del modelo para evitar el sobreajuste. En este código el parámetro de regulación Alpha varía para encontrar el mejor equilibrio entre el ajuste y la complejidad del modelo. Por otro lado, la regresión Lasso es muy similar a la regresión de Ridge. En el código se utiliza para realizar la predicción de la popularidad de las canciones, ya que al usar

esta regresión se evalúa la capacidad de selección de características, así como la capacidad de regularizar el modelo. En este caso, como con Ridge, se varía el valor del parámetro Alpha para proporcionar un buen ajuste a los datos y mejorar la precisión de la predicción de canciones. Finalmente, el modelo de Random Forest se usó para reducir el sobreajuste y mejorar la precisión del modelo. En el código se utilizan valores diferentes para n_estimators en el bosque para determinar cuál proporciona el mejor rendimiento.

- **Explicación de parámetros**

  Neg-MSE (Negative Mean Squared Error): Es una métrica utilizada para evaluar el desempeño de un modelo de regresión. MSE calcula la diferencia cuadrática media entre el valor predicho y el valor real objetivo. Los valores negativos de MSE generalmente se usan con GridSearchCV de scikit-learn, ya que intenta aumentar el resultado. Por lo tanto, un valor de MSE negativo más alto indica un mejor rendimiento del modelo.

  RMSE (Root Mean Squared Error): Es la raíz cuadrada de MSE para proporcionar una medida más interpretable del error medio. RMSE tiene las mismas unidades que la variable objetivo y representa el error estándar de las predicciones del modelo. Un valor RMSE más bajo indica que el modelo se ajusta mejor a los datos.

  Los parámetros óptimos dependen del algoritmo y los hiperparámetros establecidos. Para los modelos de regresión, los parámetros comunes incluyen valores alfa para Ridge y Lasso, y el número de estimadores (n_estimators) para bosques aleatorios. Los mejores parámetros encontrados representan la configuración que maximiza el rendimiento del modelo según la métrica de evaluación elegida (MSE negativo en este caso).

- **Conclusiones**

- Se resalta la importancia de un *dataset* que esté limpio y que tenga los datos de forma correcta, porque es esencial tener todo acondicionado para poder hacer un análisis óptimo del problema a resolver.
- Los modelos de regresión lineal, Ridge y Lasso tuvieron resultados similares para MSE y RMSE negativos. Esto muestra que estos modelos tienen un rendimiento comparable en la predicción de la popularidad de las canciones.
- El modelo de Lasso con regularización L1 funciona ligeramente peor que los otros modelos. Esto sugiere que la regularización L1 puede no ser apropiada para este conjunto de datos en particular.
- En cuanto a los parámetros óptimos, el modelo Ridge muestra un mejor desempeño con un valor de alfa igual a 10.0, lo que indica que la regularización de L2 es beneficiosa en este caso.
- El modelo Random Forest fue el que presentó el menor neg-MSE y RMSE, lo cual indica que tiene un mejor rendimiento en la predicción de la popularidad de canciones en Spotify en comparación con los otros modelos.

- Se encontraron los mejores parámetros para el modelo Ridge (alpha: 10.0), el modelo Lasso (alpha: 0.1) y el modelo Random Forest (n_estimators: 200). Estos resultados proporcionan información valiosa para seleccionar el mejor modelo y configuración de parámetros para futuras predicciones de popularidad de canciones en Spotify.