Actividad IA

Robert Steven Delgado Peralta

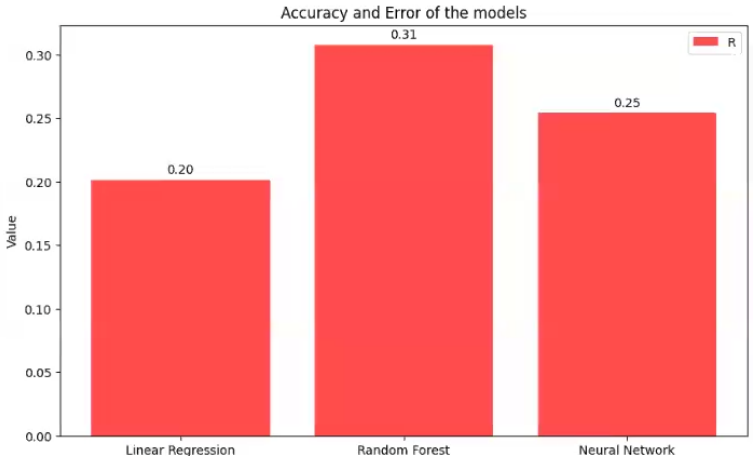
Desarrollo de Software – Parquesoft Ti

Actividad 1

Titulo:	Prediction of music genre
Resumen:	Conjunto de datos de canciones con valores que miden características musicales tales como su duración, ser bailables, acústicas, enérgicas, instrumentales, ruidosas, entre otras. Así también, el conjunto de datos cuenta con valores, cualitativos como lo son su tempo, la tónica y el género de la canción, además, tiene datos de identificación como el id, el artista y el nombre de la canción. También cuenta con un dato de interés el cual es la popularidad de la canción. ***Para esta actividad, se intentará medir la popularidad de una canción de acuerdo con las características de esta.***
Origen:	https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre/data
Número total de Variables:	18
Número total de Variables Cualitativas:	6
Número total de Variables Cuantitativas:	12
Variable por Predecir:	Popularity
Algoritmo de Predicción:	Comparación entre: LinearRegression, RandomForestRegressor, MLPRegressor

Actividad 2

En la clase del 9 de julio, se presentó un avance de la actividad con los resultados de la comparación de los tres modelos (donde se presume el valor de r^2 como precisión) y el proceso realizado hasta ese momento:



De la revisión del proceso por parte del profesor Miguel Orozco se obtuvo la siguiente retroalimentación:

- Normalizar los datos cuantitativos ya que los valores distan en ordenes de 10^5 entre algunos de los datos
- Agregar el genero clasificado de forma numérica

Así las cosas, se modifico el proceso realizado incluyendo también la clasificación del tempo de acuerdo con los tipos de tempos establecidos en la música *(Largo, Lento, Adagio, Andante, Moderato, Allegro, Presto)*. A continuación, se presenta el proceso realizado con dichas modificaciones y los resultados obtenidos de este.

Importación de librerías

Analisis de Modelos.py M

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score
from sklearn.preprocessing import MinMaxScaler
```

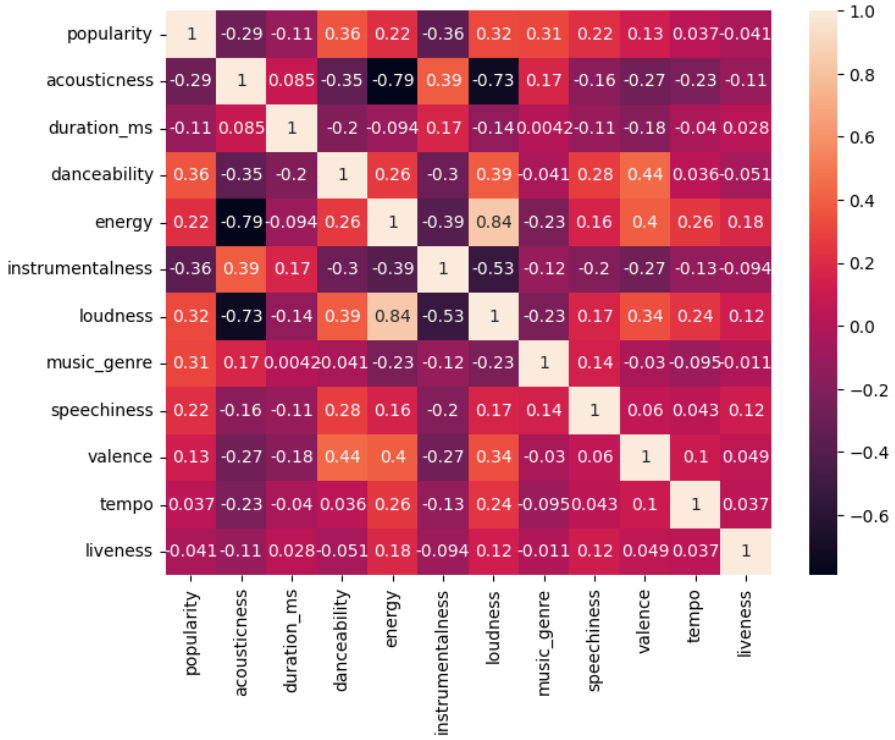
Carga de datos y preparación de las variables

```python
# Data transformation and loading


# Function that classify the tempo colum according to the normal tempo ranges
def tempoClassifier(x):
    if (x <= 40):
        x = 1  # Largo
    elif (x <= 66):
        x = 2  # Lento
    elif (x <= 76):
        x = 3  # Adagio
    elif (x <= 108):
        x = 4  # Andante
    elif (x <= 120):
        x = 5  # Moderato
    elif (x <= 168):
        x = 6  # Allegro
    else:
        x = 7  # Presto
    return x


dataMusic = pd.read_csv('./music_genre.csv')
dataMusic = dataMusic.dropna()

# Drop rows with duration values lees or equal to 0
dataMusic = dataMusic.drop(dataMusic[(dataMusic['duration_ms'] <= 0)].index)

# Drop rows with tempo values equals to ?
dataMusic = dataMusic.drop(dataMusic[(dataMusic['tempo'] == '?')].index)

# Change the type of tempo column
dataMusic['tempo'] = dataMusic['tempo'].astype('float64')

# Drop rows with tempo values lees or equal to 0
dataMusic = dataMusic.drop(dataMusic[(dataMusic['tempo'] <= 0)].index)

inlets = dataMusic[['acousticness', 'duration_ms',
                    'danceability', 'energy', 'instrumentalness', 'loudness', 'music_genre', 'speechiness', 'valence', 'tempo', 'liveness']]

# Diccionary to change the music_genere value from string to integer
genres = {'Electronic': 1, 'Anime': 2, 'Jazz': 3, 'Alternative': 4,
          'Country': 5, 'Rap': 6, 'Blues': 7, 'Rock': 8, 'Classical': 9, 'Hip-Hop': 10}
inlets['music_genre'] = inlets['music_genre'].map(genres)
```

```python
# Classification of the tempo column
inlets['tempo'] = inlets['tempo'].apply(
    lambda x: tempoClassifier(x))

# Normalisation of the inlets data
transformer = MinMaxScaler()
inlets = transformer.fit_transform(inlets)
inlets = pd.DataFrame(inlets, columns=['acousticness', 'duration_ms',
                                       'danceability', 'energy', 'instrumentalness', 'loudness', 'music_genre', 'speechiness', 'valence', 'tempo', 'liveness'])

oulets = dataMusic['popularity']

x_train, x_test, y_train, y_test = train_test_split(
    inlets, oulets, test_size=0.2, random_state=42)
```

Se realizo el análisis de la matriz de correlación para verificar los valores, acotando esta solo a las características que se utilizaron en el modelo.

```python
# Correlation Matrix
modelValues = dataMusic[['popularity', 'acousticness', 'duration_ms',
                         'danceability', 'energy', 'instrumentalness', 'loudness', 'music_genre', 'speechiness', 'valence', 'tempo', 'liveness']]
modelValues['music_genre'] = modelValues['music_genre'].map(genres)
modelValues['tempo'] = modelValues['tempo'].apply(
    lambda x: tempoClassifier(x))
modelValues = transformer.fit_transform(modelValues)
modelValues = pd.DataFrame(modelValues, columns=['popularity', 'acousticness', 'duration_ms',
                                                 'danceability', 'energy', 'instrumentalness', 'loudness', 'music_genre', 'speechiness', 'valence', 'tempo', 'liveness'])
corrData = modelValues.corr(method="pearson")
plt.figure(figsize=(8, 6))
sns.heatmap(corrData, annot=True)
plt.show()
```

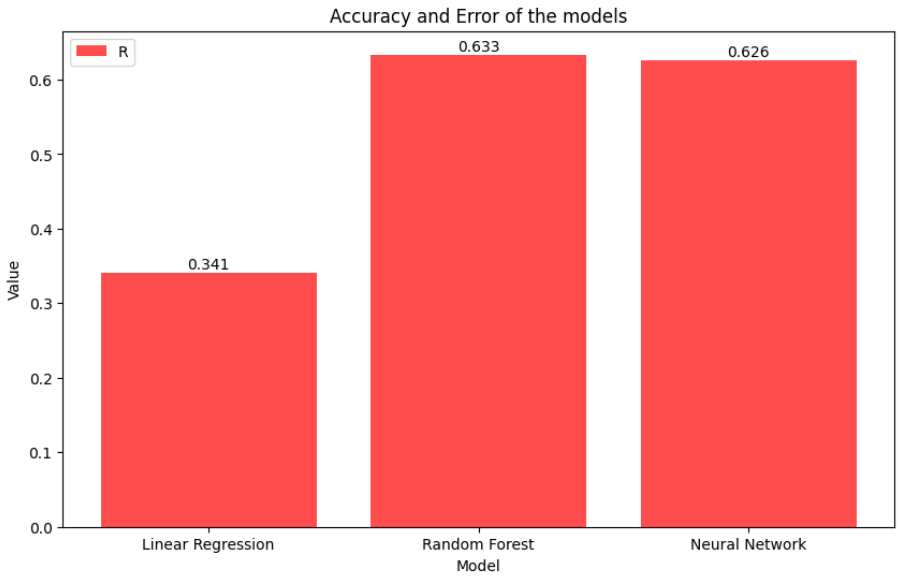


Entrenamiento de los modelos y predicción de la variable

```
# Linear Regression model
linearModel = LinearRegression()
linearModel.fit(x_train, y_train)

# Random Forest Regressor model
randomForestModel = RandomForestRegressor(n_estimators=199, random_state=42)
randomForestModel.fit(x_train, y_train)

# Neural Network model with Scikit-Learn
neuralNetworkModel = MLPRegressor(hidden_layer_sizes=(128, 64), activation=(
    'relu'), max_iter=500, random_state=42, solver='adam', learning_rate_init=0.01)
neuralNetworkModel.fit(x_train, y_train)

# Predictions in all the models
yPredLinear = linearModel.predict(x_test)
yPredRandomForest = randomForestModel.predict(x_test)
yPredNeuralNetwork = neuralNetworkModel.predict(x_test)

# Calculation of the Coefficient of Determination (R Squared) for all the models
r2Linear = r2_score(y_test, yPredLinear) * 100
r2RandomForest = r2_score(y_test, yPredRandomForest) * 100
r2NeuralNetwork = r2_score(y_test, yPredNeuralNetwork) * 100
```

Resultados

```
# Create Bar Graphics in order to evaluate the accuracy of the models
models = ['Linear Regression', 'Random Forest', 'Neural Network']
r2Scores = [r2Linear, r2RandomForest, r2NeuralNetwork]

# R Squared and MSE Graphic
plt.figure(figsize=(10, 6))
bars = plt.bar(models, r2Scores, color='red', alpha=0.7)
for bar, r2 in zip(bars, r2Scores):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() +
             0.005, f'{r2:.2f}%', ha='center', color='black')

plt.title('Accuracy and Error of the models')
plt.xlabel('Model')
plt.ylabel('Value')
plt.legend('R_Squared')
plt.show()

```



Como se puede apreciar en esta ultima imagen, el incremento del valor de r^2 es considerable en todos los modelos con las modificaciones realizadas en el procedimiento. Si bien al inicio del proyecto se esperaba obtener valores mayores para esta métrica, el incremento en los resultados entre la primera versión del procedimiento y el planteamiento final da cumplimiento al objetivo de realizar procesos de acercamiento a la inteligencia artificial y los procedimiento y librerías que se utilizan con esta.

Enlace del repositorio de la actividad: https://github.com/DrowAnn/Introduction-to-AI-Activity