

# ACTIVIDAD IA

Carol Marcela Ramirez Alvarado

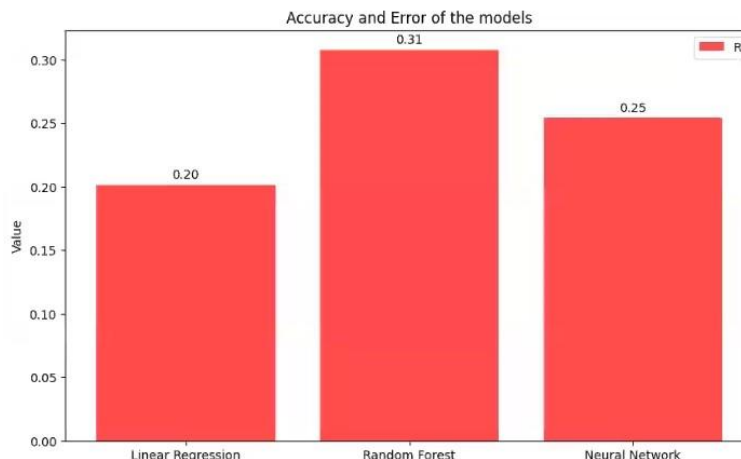
Desarrollo de Software – Parquesoft TI

## Actividad 1

<b>Título:</b>	Prediction of music genre
<b>Resumen:</b>	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. <b><i>Para esta actividad, se intentará medir la popularidad de una canción de acuerdo con las características de esta.</i></b>
<b>Origen:</b>	<a href="https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre/data">https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre/data</a>
<b>Número total de Variables:</b>	18
<b>Número total de Variables Cualitativas:</b>	6
<b>Número total de Variables Cuantitativas:</b>	12
<b>Variable por Predecir:</b>	Popularity
<b>Algoritmo de Predicción:</b>	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 modificó 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

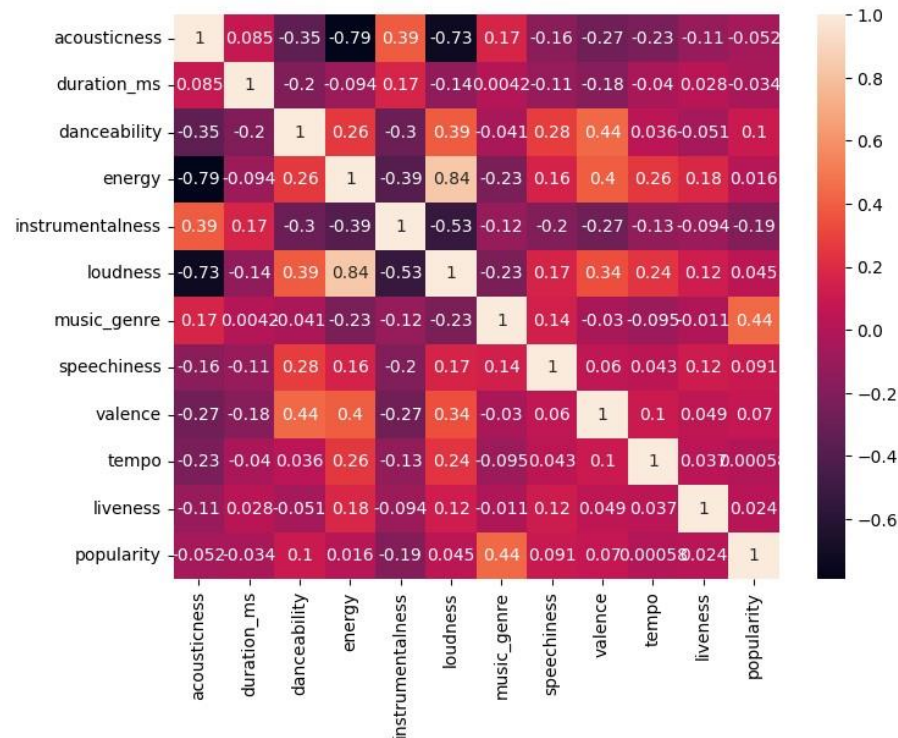
```
Analisis de Modelos.py M X
1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LinearRegression
6 from sklearn.ensemble import RandomForestRegressor
7 from sklearn.neural_network import MLPRegressor
8 from sklearn.metrics import r2_score
9 from sklearn.preprocessing import MinMaxScaler
```

## Carga de datos y preparación de las variables

```
11 # Data transformation and loading
12
13
14 # Function that classify the tempo colum according to the normal tempo ranges
15 def tempoClassifier(x):
16     if (x <= 40):
17         x = 1 # Largo
18     elif (x <= 66):
19         x = 2 # Lento
20     elif (x <= 76):
21         x = 3 # Adagio
22     elif (x <= 108):
23         x = 4 # Andante
24     elif (x <= 120):
25         x = 5 # Moderato
26     elif (x <= 168):
27         x = 6 # Allegro
28     else:
29         x = 7 # Presto
30     return x
31
32
33 dataMusic = pd.read_csv('./music_genre.csv')
34 dataMusic = dataMusic.dropna()
35
36 # Drop rows with duration values lees or equal to 0
37 dataMusic = dataMusic.drop(dataMusic[(dataMusic['duration_ms'] <= 0)].index)
38
39 # Drop rows with tempo values equals to ?
40 dataMusic = dataMusic.drop(dataMusic[(dataMusic['tempo'] == '?')].index)
41
42 # Change the type of tempo column
43 dataMusic['tempo'] = dataMusic['tempo'].astype('float64')
44
45 # Drop rows with tempo values lees or equal to 0
46 dataMusic = dataMusic.drop(dataMusic[(dataMusic['tempo'] <= 0)].index)
47
48 inlets = dataMusic[['acousticness', 'duration_ms',
49                    'danceability', 'energy', 'instrumentalness', 'loudness', 'music_genre', 'speechiness', 'valence', 'tempo', 'liveness']]
50
51 # Diccionary to change the music_genre value from string to integer
52 genres = {'Electronic': 1, 'Anime': 2, 'Jazz': 3, 'Alternative': 4,
53           'Country': 5, 'Rap': 6, 'Blues': 7, 'Rock': 8, 'Classical': 9, 'Hip-Hop': 10}
54 inlets['music_genre'] = inlets['music_genre'].map(genres)
55
56 # Classification of the tempo column
57 inlets['tempo'] = inlets['tempo'].apply(
58     lambda x: tempoClassifier(x))
59
60 # Normalisation of the inlets data
61 transformer = MinMaxScaler()
62 inlets = transformer.fit_transform(inlets)
63 inlets = pd.DataFrame(inlets, columns=['acousticness', 'duration_ms',
64                                     'danceability', 'energy', 'instrumentalness', 'loudness', 'music_genre', 'speechiness', 'valence', 'tempo', 'liveness'])
65
66 outlets = dataMusic['popularity']
67
68 x_train, x_test, y_train, y_test = train_test_split(
69     inlets, outlets, 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.

```
71 # Correlation Matrix
72 modelValues = inlets
73 modelValues['popularity'] = outlets
74 corr_df = modelValues.corr(method="pearson")
75 plt.figure(figsize=(8, 6))
76 sns.heatmap(corr_df, annot=True)
77 plt.show()
```



### Entrenamiento de los modelos y predicción de la variable

```

79 # Linear Regression model
80 linearModel = LinearRegression()
81 linearModel.fit(x_train, y_train)
82
83 # Random Forest Regressor model
84 randomForestModel = RandomForestRegressor(n_estimators=199, random_state=42)
85 randomForestModel.fit(x_train, y_train)
86
87 # Neural Network model with Scikit-Learn
88 neuralNetworkModel = MLPRegressor(hidden_layer_sizes=(128, 64), activation=(
89     'relu'), max_iter=500, random_state=42, solver='adam', learning_rate_init=0.01)
90 neuralNetworkModel.fit(x_train, y_train)
91
92 # Predictions in all the models
93 yPredLinear = linearModel.predict(x_test)
94 yPredRandomForest = randomForestModel.predict(x_test)
95 yPredNeuralNetwork = neuralNetworkModel.predict(x_test)
96
97 # Calculation of the Coefficient of Determination (R Squared) for all the models
98 r2Linear = r2_score(y_test, yPredLinear) * 100
99 r2RandomForest = r2_score(y_test, yPredRandomForest) * 100
100 r2NeuralNetwork = r2_score(y_test, yPredNeuralNetwork) * 100

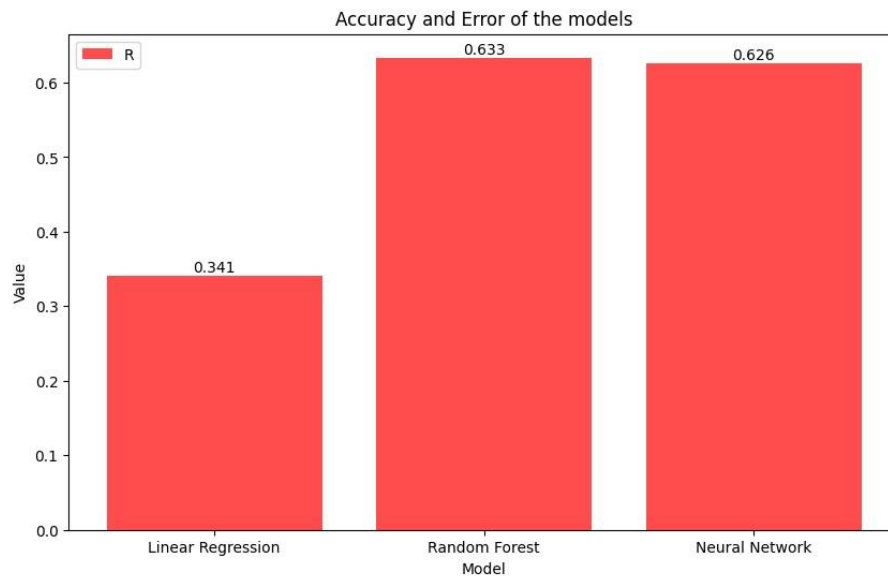
```

### Resultados

```

102 # Create Bar Graphics in order to evaluate the accuracy of the models
103 models = ['Linear Regression', 'Random Forest', 'Neural Network']
104 r2Scores = [r2Linear, r2RandomForest, r2NeuralNetwork]
105
106 # R Squared and MSE Graphic
107 plt.figure(figsize=(10, 6))
108 bars = plt.bar(models, r2Scores, color='red', alpha=0.7)
109 for bar, r2 in zip(bars, r2Scores):
110     plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() +
111             0.005, f'{r2:.2f}%', ha='center', color='black')
112
113 plt.title('Accuracy and Error of the models')
114 plt.xlabel('Model')
115 plt.ylabel('Value')
116 plt.legend('R_Squared')
117 plt.show()
118

```



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 procedimientos y librerías que se utilizan con esta.

Enlace del repositorio de la actividad: [https://github.com/CarolMarcela/Entrega\\_Inteligencia\\_Artificial.git](https://github.com/CarolMarcela/Entrega_Inteligencia_Artificial.git)