Instituto Politécnico Nacional

Escuela Superior de Cómputo

“Inteligencia Artificial”

Proyecto Final

Agente Inteligente

Profesor:

Efrén Cruz Carlín

Alumnos:

León Flores Daniela Alejandra

López Gabriel Efraín

Fecha de presentación: 5 de enero del 2026

Introducción

En la actualidad, la Inteligencia Artificial (IA) se ha convertido en una herramienta fundamental para el análisis y procesamiento de grandes volúmenes de información, permitiendo la identificación de patrones complejos en distintos ámbitos. Uno de estos ámbitos es la industria musical, donde el análisis de características acústicas ha permitido desarrollar sistemas capaces de clasificar, recomendar y analizar canciones de manera automatizada.

El presente proyecto consiste en el desarrollo de un agente inteligente para la clasificación de emociones musicales, utilizando técnicas de aprendizaje automático supervisado. El sistema se apoya en un conjunto de datos reales obtenidos de la plataforma Spotify, el cual contiene información acústica y emocional de diversas canciones. A partir de estos datos, se entrena un modelo capaz de predecir la emoción predominante de una canción seleccionada por el usuario.

El agente fue implementado mediante un entorno web interactivo, que permite al usuario navegar de forma jerárquica a través de géneros y subgéneros musicales, seleccionar una canción específica y obtener como resultado la emoción estimada por el modelo. Para la clasificación se empleó el algoritmo K-Nearest Neighbors (KNN), debido a su simplicidad, interpretabilidad y adecuación para trabajar con características numéricas normalizadas.

Este proyecto tiene como objetivo demostrar la aplicación práctica de los conceptos vistos en clase, integrando el uso de un dataset real, un algoritmo de aprendizaje automático y una interfaz web funcional, cumpliendo con los criterios establecidos en la rúbrica de evaluación para el desarrollo de agentes inteligentes.

Descripción del agente

El agente inteligente desarrollado corresponde a un sistema de clasificación de emociones musicales, cuyo objetivo es predecir la emoción asociada a una canción a partir de sus características acústicas. El agente permite al usuario interactuar mediante una interfaz web, seleccionando de forma jerárquica un género musical, un subgénero y posteriormente una canción, sobre la cual el sistema aplica un modelo de aprendizaje automático para determinar la emoción predominante.

El agente actúa de manera reactiva, ya que responde a las decisiones del usuario y genera una predicción inmediata sin modificar su estructura interna durante la ejecución.

Tipo y clasificación de IA

El sistema implementa un agente basado en Aprendizaje Automático Supervisado. Se clasifica como:

- Tipo de IA: Inteligencia Artificial Débil (IA específica)
- Paradigma: Aprendizaje Automático
- Tipo de aprendizaje: Supervisado
- Tipo de agente: Agente reactivo con percepción limitada

El modelo aprende a partir de un conjunto de datos etiquetados, donde cada canción cuenta con una emoción previamente definida, permitiendo que el agente generalice patrones para nuevas predicciones.

Medio ambiente de trabajo

El desarrollo del agente se realizó en un entorno local utilizando las siguientes herramientas y tecnologías:

- Lenguaje de programación: Python 3.12
- Framework web: Flask
- Librerías principales:
    - Pandas (manipulación de datos)
    - Scikit-learn (modelado y aprendizaje automático)
    - NumPy (operaciones numéricas)
- Entorno de desarrollo: Visual Studio Code

- Interfaz: Navegador web (localhost)

El sistema se ejecuta localmente y es accesible mediante un navegador, lo que facilita su uso y evaluación.

Modelo de datos y características

El agente utiliza un dataset musical de Spotify obtenido desde Kaggle, el cual contiene información acústica, artística y emocional de múltiples canciones.

Para el modelo de clasificación se seleccionaron únicamente características numéricas relevantes, evitando atributos textuales que no pueden ser procesados directamente por el algoritmo.

Características utilizadas:

- Tempo
- Energy
- Danceability
- Positiveness
- Speechiness
- Liveness
- Acousticness
- Instrumentalness

La variable objetivo (label) es:

- emotion → emoción asociada a la canción (joy, sadness, anger, fear, etc.)

Antes del entrenamiento:

- Se eliminaron valores nulos
- Se normalizaron las características mediante StandardScaler
- Se filtraron los datos por género y subgénero para mejorar la coherencia semántica

Algoritmo usado

El algoritmo seleccionado es K-Nearest Neighbors (KNN), un método de clasificación supervisada basado en distancia.

Justificación del uso de KNN:

- Es adecuado para datos numéricos normalizados
- Permite interpretar similitud entre canciones
- No requiere una fase compleja de entrenamiento
- Funciona correctamente para clasificación multiclase

Configuración utilizada:

- Número de vecinos: k = 7
- Métrica de distancia: Euclidiana

El modelo predice la emoción de una canción analizando las canciones más cercanas en el espacio de características acústicas.

Explicación del código

1. ***Limpieza y transformación de datos***

```
def time_to_seconds(t):
try:
t = str(t)
if ":" in t:
m, s = t.split(":")
return int(m) * 60 + int(s)
else:
return float(t)
except:
return np.nan
```

Debido a que la duración de las canciones se encuentra en formato de tiempo (mm:ss), se define una función que convierte este valor a segundos, permitiendo su uso como una característica numérica dentro del modelo.

```
df["Length"] =
df["Length"].apply(time_to_seconds)
```

Posteriormente, se aplica esta función a la columna Length del dataset.

```
df["Loudness (db)"] = (
    df["Loudness (db)"]
    .astype(str)
    .str.replace("db", "", regex=False)
    .astype(float)
)
```

La columna Loudness (db) se transforma eliminando la cadena “db” y convirtiendo el valor a tipo numérico, asegurando que pueda ser procesada por el algoritmo de aprendizaje automático.

2. ***Selección de características***

```
features = [
"Length",
"Tempo",
"Loudness (db)",
"Energy",
"Danceability",
"Positiveness",
"Speechiness",
"Liveness",
"Acousticness",
"Instrumentalness",
"Similarity Score 1",
"Similarity Score 2",
"Similarity Score 3"
 ]
```

Se seleccionan únicamente las características numéricas relevantes para el modelo, relacionadas con propiedades acústicas y de similitud entre canciones. La variable objetivo es la columna emotion.

```
df_model = df[features + ["emotion"]].dropna()
```

Se eliminan los registros con valores nulos para garantizar la consistencia del entrenamiento.

3. ***Separación de variables independientes y dependientes***

```
X = df_model[features]
y = df_model["emotion"]
```

Las variables independientes (X) corresponden a las características acústicas, mientras que la variable dependiente (y) representa la emoción asociada a cada canción.

4. ***Escalado de datos***

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Se utiliza StandardScaler para normalizar los datos, lo cual es necesario para el correcto funcionamiento del algoritmo KNN, ya que este se basa en distancias entre puntos.

5. ***División del conjunto de datos***

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled,
    y,
    test_size=0.2,
    random_state=42
)
```

El dataset se divide en un conjunto de entrenamiento (80 %) y uno de prueba (20 %), permitiendo evaluar el desempeño del modelo y evitar sobreajuste.

6. ***Entrenamiento del modelo KNN***

```
knn = KNeighborsClassifier(
    n_neighbors=7,
    metric="euclidean"
)
knn.fit(X_train, y_train)
```

Se implementa el algoritmo **K-Nearest Neighbors**, configurado con 7 vecinos y distancia euclidiana. El modelo se entrena utilizando los datos previamente escalados.

7. ***Implementación de la aplicación web***

```
app = Flask(__name__)
```

Se inicializa la aplicación web utilizando Flask.

```
@app.route("/", methods=["GET", "POST"])
def index():
```

Se define la ruta principal, la cual gestiona tanto la visualización del formulario como el procesamiento de los datos ingresados por el usuario.

```
datos = [
    float(request.form["Length"]),
    float(request.form["Tempo"]),
    ...
]
```

Los valores ingresados desde la interfaz web son recuperados y convertidos a tipo numérico para su posterior procesamiento.

```
datos_scaled = scaler.transform(datos)
emocion = knn.predict(datos_scaled)[0]
```

Los datos se normalizan utilizando el mismo escalador del entrenamiento y se aplica el modelo KNN para predecir la emoción de la canción.

8. ***Ejecución del servidor***

```
if __name__ == "__main__":
    app.run(debug=True)
```

Conclusión

Con este proyecto logramos implementar de manera satisfactoria un agente inteligente para la clasificación de emociones musicales, integrando conceptos fundamentales de la Inteligencia Artificial, el Aprendizaje Automático y el desarrollo de aplicaciones web. A partir de un dataset basado en Spotify obtenido por la página de Kaggle, nos fue posible procesar y normalizar la información acústica relevante para entrenar un modelo capaz de identificar patrones emocionales en las canciones.

El uso del algoritmo KNN (K-Nearest Neighbors) nos permitió realizar la clasificación de emociones de forma eficiente, aprovechando la similitud entre canciones mediante características numéricas previamente escaladas. De igual forma, la correcta limpieza y transformación de los datos fue un factor clave para garantizar un buen funcionamiento del modelo y la fiabilidad de las predicciones generadas.

Por otro lado, la implementación de una interfaz web interactiva nos facilitó la interacción con el usuario, permitiendo seleccionar géneros, subgéneros y canciones de manera intuitiva, y visualizar de forma clara la emoción predicha por el agente. Esta integración entre el modelo de aprendizaje automático y el entorno web demostró la aplicabilidad práctica de los sistemas inteligentes en escenarios reales.

Finalmente, el proyecto cumple con los criterios establecidos en la rúbrica de evaluación, al presentar un agente funcional, un uso adecuado de técnicas de Inteligencia Artificial, un correcto aprovechamiento del dataset y una interfaz accesible.