INTRODUCCIÓN

PROYECTO DE MACHINE LEARNING

- REALIZADO POR: IVAN GONZALO TAPIA
- INSTITUCIÓN: UNIVERSIDAD PROVINCIAL DEL SUDESTE
- MATERIA: BIG DATA Y APRENDIZAJE DE MÁQUINA
- DOCENTE: ING. VALENTÍN BARCO
- LUGAR: PUNTA ALTA
- FECHA: 08 DE NOVIEMBRE DE 2023
- REPOSITORIO PROYECTO: GITHUB PROYECTO 2 MACHINE LEARNING
- INFORME: INFORME EN WORD
- NOTEBOOK: COLAB

JUSTIFICACIÓN DEL INFORME Y PROPÓSITOS

SE DEBE DESARROLLAR UN PROGRAMA QUE CATEGORICE CANCIONES SEGÚN LAS PREFERENCIAS DEL USUARIO. LA TAREA PRINCIPAL ES CREAR UN PROGRAMA CAPAZ DE PREDECIR SI UNA CANCIÓN PODRÍA SER DEL GUSTO DEL USUARIO O NO. ESTE PROYECTO DEBE SER EJECUTADO DENTRO DE UN NOTEBOOK DE COLAB.

1- IMPORTACIÓN DE LIBRERIAS, CARGA Y PREPROCESAMIENTO DE DATOS.
2- VISUALIZACIÓN DE DATOS CON MATPLOTLIB Y SEABORN.
3- FUNCIÓN PARA OBSERVAR LAS MÉTRICAS Y EVALUAR LOS MODELOS.
4- CREACIÓN DE MODELOS DE ML CON SKLEARN.
5- CREACIÓN DE MODELOS CON TENSOR FLOW KERAS.
6- SELECCIÓN Y AJUSTE HIPERPARAMETROS DE LOS MEJORES MODELOS.
7- ENSAMBLE DE LOS MEJORES MODELOS POR VOTACIÓN MAYORITARIA Y EVALUACIÓN DEL MISMO.
8- EXPLORANDO OTRAS LIBRERÍAS DE MACHINE LEARNING COMO XGBOOST, LIGHTGBM, CATBOOST.

1. Importación de Librerías, Carga de Datos y Preprocesamiento de Datos.

Librerías Necesarias

Para iniciar el proyecto, importamos las siguientes librerías esenciales:

os para operaciones del sistema

- **Pandas** para el manejo de datos
- **Numpy** para cálculos numéricos
- **Matplotlib** y S**eaborn** para la visualización de datos
- **Sklearn** para el aprendizaje automático
- **preprocessing** de S**klearn** para la codificación de etiquetas

Importamos los datos desde el archivo CSV 'Canciones_Spotify.csv' y creamos un DataSet para su análisis.
Preprocesamiento de Datos

En esta etapa, realizamos las siguientes transformaciones:

- **Selección de Variables de Entrada:** Elegimos atributos relevantes, como 'acousticness', 'danceability', 'energy', 'instrumentalness', 'speechiness', 'valence' y 'tempo', que consideramos importantes para la predicción, como así tambien creamos diferentes variables entre estas para hacer diferentes X para el modelo y que estas diferentes X, tengan distintos tipos de variables.
- **Split de Datos:** Dividimos los datos en conjuntos de entrenamiento y prueba utilizando la función **train_test_split** de **sklearn,** dejando un 40% de los datos para testeo.
- **Procesamiento de Datos:** Utilizamos la codificación de etiquetas (Label Encoding de SKLearn) para convertir las etiquetas en valores numéricos, si bien las variables que utilizamos no necesitan un estricto procesamiento, lo realizamos como buenas prácticas para cuando se crean modelos de Machine Learning.

2. Visualización de Datos con Seaborn y Matplotlib.

En esta sección, utilizamos visualización para comprender mejor nuestros datos.

Empleamos las bibliotecas Seaborn y Matplotlib para generar un gráfico Pairplot, que nos permite visualizar la relación entre todas las variables del conjunto de datos.

3. Función para observar las métricas de medición y evaluar los modelos.

Hemos definido una función llamada "evaluar_modelo" que evalúa el rendimiento de los modelos. Calcula métricas clave, como la precisión, la recuperación y el puntaje F1, y crea una matriz de confusión para visualizar las predicciones del modelo en comparación con los valores reales. Además de la evaluación simple del modelo, implementamos la validación cruzada, que nos permite evaluar el rendimiento del modelo en múltiples particiones del conjunto de datos de entrenamiento.

4. Creación de modelos de clasificación de ml con sklearn.

Los modelos de clasificación son herramientas que encontramos dentro del aprendizaje automático, nos permiten asignar una etiqueta o categoría a un conjunto de datos en función de sus características.

En este proyecto, aplicamos varios modelos de clasificación para determinar si a un usuario le gusta o no una canción en función de sus atributos musicales. A continuación, se describen algunos de los modelos utilizados junto con una breve explicación de su funcionamiento:

4.1 K-Nearest Neighbors (KNN)

El modelo K-Nearest Neighbors (KNN) clasifica nuevos datos basándose en la mayoría de las etiquetas de sus vecinos más cercanos en un espacio multidimensional. En este caso, KNN compara una canción con las canciones vecinas en un espacio de características y asigna una etiqueta de "gusto" o "no gusto" en función de las canciones más similares.

4.2- Support Vector Machine (SVM)

El modelo Support Vector Machine (SVM) busca encontrar un hiperplano de separación óptimo en un espacio multidimensional. En este proyecto, SVM se utiliza para clasificar canciones en dos categorías: "gusto" y "no gusto" basándose en las características musicales. Es especialmente eficaz cuando las clases no son linealmente separables.

4.3- Decision Tree (Árbol de Decisión)

El modelo Decision Tree divide el conjunto de datos en subconjuntos más pequeños basados en reglas condicionales. En este proyecto, clasifica canciones en función de sus características, tomando decisiones basadas en estas características y dividiendo el conjunto de datos en categorías más pequeñas hasta obtener una clasificación final.

4.4- Naive Bayes

El modelo Naive Bayes es un algoritmo de clasificación basado en el teorema de Bayes con la suposición ingenua de independencia condicional entre las características. Aunque esta suposición es simplificada, Naive Bayes es efectivo en la clasificación de datos, especialmente en tareas de procesamiento de lenguaje natural y filtrado de spam.

4.5- Random Forest

El modelo Random Forest se basa en la construcción de múltiples árboles de decisión durante el proceso de entrenamiento. Cada árbol toma decisiones de clasificación y la predicción final se obtiene mediante la votación o el promedio de las predicciones de los árboles. Es conocido por su capacidad para manejar conjuntos de datos grandes y variables de entrada diversificadas.

4.6- Logistic Regression (Regresión Logística)

A pesar de su nombre, la Regresión Logística se utiliza para predecir la probabilidad de que una instancia pertenezca a una de las dos clases posibles. Se basa en la función logística para transformar valores de entrada en un rango de 0 a 1, representando probabilidades.

4.7- GradientBoostingClassifier

El GradientBoostingClassifier es un algoritmo de aprendizaje automático que utiliza árboles de decisión débiles. Entrena árboles secuencialmente y ajusta cada árbol a los errores del modelo anterior, mejorando gradualmente el rendimiento.

4.8- MLPClassifier (Multilayer Perceptron Classifier)

El MLPClassifier es parte de las redes neuronales artificiales y se utiliza en problemas de clasificación. Está compuesto por múltiples capas de neuronas interconectadas y es eficaz en problemas de alta dimensionalidad y no lineales.

4.9- AdaBoostClassifier (Adaptive Boosting Classifier)

El AdaBoostClassifier combina clasificadores débiles en un clasificador fuerte. En cada iteración, asigna pesos a las instancias de datos y ajusta un clasificador débil, enfocándose en las instancias que se clasificaron incorrectamente en las iteraciones anteriores.

4.10- Support Vector Classifier (SVC)

El Support Vector Classifier (SVC) busca encontrar un hiperplano óptimo que maximice el margen entre las clases en el espacio de características. Es eficaz en problemas de clasificación binaria y multiclase, especialmente cuando las clases no son linealmente separables.

5. Creación de modelos con tensor flow keras.

En este paso, desarrollamos modelos de red neuronal utilizando las bibliotecas TensorFlow y Keras. Las redes neuronales son una técnica poderosa en el aprendizaje profundo.

El modelo de red neuronal se ha definido como un modelo secuencial, lo que significa que las capas de la red se apilan una tras otra. En una red neuronal, las capas se componen de neuronas interconectadas que procesan información. Cada capa toma la salida de la capa anterior y realiza cálculos en ella.

En el contexto de la clasificación de preferencias musicales, estas redes neuronales utilizan las características musicales de las canciones como entrada y, a medida que avanzan a través de las capas, aprenden a reconocer patrones y características relevantes para la tarea de clasificación. Las capas intermedias de la red, conocidas como capas ocultas, realizan transformaciones matemáticas que permiten a la red aprender representaciones cada vez más abstractas de los datos.

Finalmente, la última capa de la red produce una salida que representa la probabilidad de que una canción sea clasificada como "gusto" o "no gusto". En otras palabras, la red neuronal estima la probabilidad de pertenecer a una de las dos categorías.

Sin embargo, es importante destacar que, en este proyecto en particular, los modelos de redes neuronales no resultaron ser la solución más útil para el problema de clasificación de preferencias musicales. A pesar de su capacidad para aprender representaciones complejas de datos, el rendimiento de estos modelos puede verse limitado por la cantidad de datos y la complejidad del problema. En comparación con otros modelos de clasificación, las redes neuronales no lograron superar significativamente su rendimiento. Por lo tanto, en este

contexto específico, se optó por enfocarse en otros modelos de aprendizaje automático que proporcionaron resultados más sólidos y eficaces.

6. Selección y ajuste hiperparametros de los mejores modelos.

En esta etapa del proyecto, llevamos a cabo la selección de los modelos que han demostrado un alto rendimiento en la tarea de clasificar las preferencias de los usuarios en función de las características de las canciones. La elección de estos modelos se basa en métricas de rendimiento, como precisión, recall, F1-score y exactitud (accuracy). Los modelos que han alcanzado una puntuación superior a 0.70 de accuracy se han considerado los más efectivos en esta tarea.

La razón detrás de la selección de estos modelos radica en su capacidad para capturar relaciones y patrones complejos en los datos. Dado que el problema de clasificar las preferencias de los usuarios a partir de características musicales puede ser intrincado, se requieren modelos que sean lo suficientemente flexibles y capaces de aprender estas relaciones subyacentes. Los modelos seleccionados, como Random Forest y GradientBoostingClassifier, son de los mejores para esta tarea y, por lo tanto, son candidatos para el ajuste de hiperparámetros.

El ajuste de hiperparámetros es fundamental para maximizar el rendimiento de estos modelos seleccionados. Estas son configuraciones específicas de los modelos que influyen significativamente en su capacidad predictiva. La técnica de Grid Search se ha aplicado para explorar de manera sistemática combinaciones de hiperparámetros dentro de un rango predefinido. Este enfoque permite encontrar la combinación óptima que maximiza el rendimiento del modelo.

Además, se ha utilizado Randomized Search CV, una técnica de búsqueda aleatoria, para encontrar de manera eficiente combinaciones de hiperparámetros. En lugar de evaluar todas las combinaciones posibles como en Grid Search, se seleccionan muestras aleatorias del espacio de hiperparámetros. Esto permite un ajuste más rápido y efectivo de los modelos, lo que resulta en una mejora significativa en la capacidad de generalización.

7. Ensamble de los mejores modelos por votación mayoritaria y evaluación del mismo.

En esta etapa final del proyecto, se implementa un ensamble por votación con el propósito de combinar los modelos preentrenados que han demostrado ser los más efectivos después de ajustar sus hiperparámetros. La idea detrás de este ensamble es aprovechar la diversidad de enfoques de estos modelos para mejorar aún más el rendimiento en la clasificación de preferencias de usuario en función de las características de las canciones.

Para llevar a cabo el ensamble por votación, se han realizado predicciones individuales con cada uno de los modelos seleccionados, que incluyen Decision Tree, Random Forest, Gradient Boosting y AdaBoost, en el conjunto de prueba. Cada modelo proporciona su propia predicción sobre si a un usuario le gusta o no una canción.

El proceso de ensamble por votación se ha realizado de manera manual, sin utilizar el módulo VotingClassifier de sklearn. En este proceso, se contabilizan los votos de cada modelo para

cada clase de preferencia (0, 1, 2) en función de sus predicciones. La clase que recibe la mayoría de votos se elige como la predicción final del ensamble.

Una vez obtenidas las predicciones del ensamble, se han calculado métricas de rendimiento, incluyendo accuracy, precision, recall y F1-Score, con el fin de evaluar la calidad y eficacia del modelo resultante. Estas métricas proporcionan una medida cuantitativa del rendimiento del ensamble y su capacidad para clasificar las preferencias musicales de los usuarios de manera precisa.

Además, se ha generado y visualizado una matriz de confusión que compara las predicciones del ensamble con las etiquetas reales. Esta matriz de confusión ofrece una representación gráfica de la calidad de las predicciones y permite identificar posibles áreas de mejora en el modelo.

8. Probando modelos de otras librerías

En esta etapa del proyecto, se exploraron modelos de clasificación de tres bibliotecas populares de aprendizaje automático: XGBoost, LightGBM y CatBoost. Cada una de estas bibliotecas ofrece enfoques únicos y eficientes para tareas de clasificación, y se evaluaron en la clasificación de preferencias de usuario basada en características de canciones como solicita el laboratorio.

Luego, se llevó a cabo un ensamble por votación mayoritaria que combinó los modelos de estas bibliotecas junto con otros modelos previamente entrenados. El objetivo de esta etapa es aprovechar la diversidad de enfoques y características específicas de cada biblioteca para mejorar aún más el rendimiento en la tarea de clasificación. A continuación, se presentan los resultados y las conclusiones de esta evaluación y ensamble.

Resumen y conclusión

Este proyecto se enfocó en la clasificación de preferencias musicales de usuarios de Spotify utilizando diversas técnicas de aprendizaje automático. El proceso se dividió en siete etapas clave:

Introducción: Se estableció el contexto y el objetivo del proyecto: predecir si a un usuario le gustará o no una canción en función de sus características musicales.

Importación de Librerías y Preprocesamiento de Datos: Se importaron las librerías necesarias, se cargaron los datos y se realizaron tareas de limpieza y selección de variables de entrada.

Visualización de Datos: Se utilizaron Seaborn y Matplotlib para explorar visualmente las relaciones entre las variables.

Creación de Modelos de Machine Learning: Se desarrollaron diversos modelos, como KNN, SVM, Decision Tree, Naive Bayes, Random Forest, Logistic Regression, GradientBoostingClassifier, MLPClassifier y AdaBoostClassifier.

Creación de Modelos con TensorFlow y Keras (Redes Neuronales): Se explicó cómo funcionan las redes neuronales y se implementaron modelos de aprendizaje profundo para la tarea de clasificación.

Selección y Ajuste de Hiperparámetros: Se seleccionaron los modelos más efectivos y se aplicaron técnicas de ajuste de hiperparámetros, como Grid Search y Randomized Search CV, para mejorar su rendimiento.

Ensamble por Votación de los Mejores Modelos: Se combinaron los modelos preentrenados mediante un ensamble por votación para aprovechar sus diversas fortalezas y lograr una clasificación más precisa.

Resumiendo, a lo largo del proyecto, se exploraron múltiples enfoques de aprendizaje automático y se aplicaron técnicas de ajuste de hiperparámetros y ensamble para mejorar la precisión en la clasificación de preferencias musicales. La elección de modelos y estrategias se basó en la capacidad de capturar patrones complejos en los datos y lograr una alta eficacia en la tarea de predicción.