**סיכום המודלים**

1. **מודל דירוג תמונות (Image Rating Model)**
   * **מטרה**: דירוג התמונות בשדה ה-rate על פי רמת הקושי שלהם.
   * **שיטה**: שימוש בנתוני המשתמשים (כמות זמן אימון, מספר שאלות שנענו, תוצאות מבחנים וכו') כדי לדרג את התמונות במאגר הנתונים.
2. **מודל עיבוד תמונה (Image Processing Model)**
   * **מטרה**: ניתוח ודירוג תמונות חדשות שנכנסות למערכת.
   * **שיטה**: עיבוד תמונה בעזרת מודלים מתקדמים לניתוח רמת הקושי וסיווג נכון של התמונות.
3. **מודל פידבק טקסטואלי דינמי (Dynamic Text Feedback Model)**
   * **מטרה**: יצירת פידבק טקסטואלי מותאם אישית למשתמשים.
   * **שיטה**: שימוש במודל GPT ליצירת טקסט מותאם אישית שמבוסס על ביצועי המשתמשים ומספק פידבק מפורט על נקודות חוזק וחולשה.

**שיטת עבודה**

1. **איסוף נתונים**
   * שליפת נתוני המשתמשים והתמונות ממסד הנתונים.
2. **אימון מודל דירוג תמונות**
   * שימוש בנתוני המשתמשים והתמונות כדי לאמן מודל שמדרג את התמונות על פי רמת הקושי.
3. **עיבוד תמונות חדשות**
   * כל תמונה חדשה שנכנסת למערכת עוברת עיבוד תמונה כדי לדרג אותה ולהוסיף אותה למסד הנתונים.
4. **יצירת פידבק טקסטואלי דינמי**
   * ניתוח נתוני המשתמשים ויצירת פידבק טקסטואלי מותאם אישית בעזרת מודל GPT.

**יתרונות וחסרונות**

**יתרונות**

1. **שיפור מתמשך**: המודלים ילמדו וישתפרו ככל שיצטברו יותר נתונים.
2. **התאמה אישית**: פידבק טקסטואלי דינמי מותאם אישית למשתמשים בהתאם לביצועיהם.
3. **ניתוח מעמיק**: שימוש בעיבוד תמונה לניתוח מעמיק יותר של תמונות ה-ECG.
4. **פידבק איכותי**: מודל השפה מייצר פידבק איכותי ומותאם אישית למשתמש.

**חסרונות**

1. **משאבי עיבוד**: ניתוח תמונות ודירוג דורש משאבי עיבוד גדולים, במיוחד כאשר מספר התמונות גדול.
2. **תחזוקה ועדכון**: יש לעדכן ולתחזק את המודלים כך שישארו מדויקים ויעילים.
3. **עלות**: שימוש במודל GPT-3 עלול להיות יקר אם יש הרבה משתמשים והרבה פידבק נדרש.

**סיכום והסבר של קוד המודל:**

**מבוא:**

הקוד המוצג לעיל עוסק באימון ועדכון מודל עץ החלטות (Decision Tree) המבוסס על נתונים הנשמרים במסד נתונים. המודל נבנה במטרה לסייע למשתמשים לתרגל ולאמן את עצמם על ידי דירוג תמונות והערכת רמת הקושי שלהם. במהלך כל הרצה של הקוד, המודל מתעדכן ומתאים את עצמו בהתאם לנתונים החדשים המתקבלים.

**תיאור המודל:**

המודל הנבחר הוא מודל עץ החלטות (Decision Tree Classifier) מבית sklearn. מודל זה נבחר בשל מספר יתרונות חשובים:

1. **פשטות וקלות השימוש**: מודל עץ החלטות קל להבנה ולהסבר, שכן הוא משתמש במבנה היררכי של צמתים ועלים כדי לקבל החלטות.
2. **יכולת הסבר ואינטואיטיביות**: עץ ההחלטות מאפשר להסביר בקלות את התהליך שבו הוא מקבל החלטות, זאת באמצעות התרשים הוויזואלי שהוא מייצר.
3. **יכולת התאמה לנתונים קטנים וגדולים**: מודל עץ החלטות מתאים לנתונים קטנים וגדולים כאחד, מה שהופך אותו לכלי רב-גוני.

**תהליך בניית המודל:**

הקוד מתחיל בהגדרת משתני הקונפיגורציה, כמו נתיבי מסד הנתונים, מיקום שמירת המודל והלוגים. לאחר מכן, הקוד מתחיל בטעינת המודל הקיים, אם ישנו כזה, אחרת נבנה מודל חדש.

**הסבר על כל שורה בקוד:**

1. **הגדרת ספריות ומשתנים**:

import os

import pickle

import logging

import datetime

import sqlite3

from config import DB\_PATH, MODEL\_PATH, TRACE\_LOG, ERROR\_LOG, BASE\_DIR

from sklearn.tree import DecisionTreeClassifier, export\_graphviz

from sklearn.preprocessing import LabelEncoder

import numpy as np

import graphviz

שורות אלו מייבאות את כל הספריות והמודולים הנדרשים לפעולת הקוד, כולל מודולים לניהול מסד נתונים, בניית המודל והדמיית עץ ההחלטות.

1. **הגדרת מערכת הלוגים**:

logging.basicConfig(filename=TRACE\_LOG, level=logging.INFO)

כאן מוגדר קובץ הלוג (TRACE\_LOG) שבו נשמרים הודעות המעקב (trace) כדי לתעד את תהליך הבנייה והעדכון של המודל.

1. **פונקציית טעינת המודל**:

def load\_model():

...

פונקציה זו אחראית לטעינת המודל הקיים ממערכת הקבצים. אם אין מודל קיים, היא תחזיר מודל חדש.

1. **פונקציית שמירת המודל**:

def save\_model(model, feature\_importances):

...

פונקציה זו שומרת את המודל הנוכחי יחד עם המאפיינים החשובים שלו במערכת הקבצים לשימוש עתידי.

1. **פונקציית האימון והעדכון**:

def train\_or\_update\_model():

...

פונקציה זו מבצעת את עיקר העבודה בקוד. תחילה היא יוצרת חיבור למסד הנתונים ושולפת את הנתונים הרלוונטיים לאימון המודל. לאחר מכן היא מכינה את מערך התכונות והתגיות, מאמנת את המודל ומשווה את המאפיינים החשובים (feature importances) למאפיינים מהמודל הקודם. לבסוף, הפונקציה שומרת את המודל המעודכן ומייצרת הדמיה של עץ ההחלטות.

1. **הסבר על תהליך האימון**:

model.fit(X, y)

שורה זו מבצעת את תהליך האימון של המודל על הנתונים שהתקבלו, שם X מייצג את מערך התכונות ו-y את התגיות (ratings) המתאימות.

1. **שמירת המודל המעודכן**:

save\_model(model, model.feature\_importances\_)

כאן המודל נשמר יחד עם החשיבות של כל תכונה במודל.

1. **השוואת החשיבות של התכונות**:

if previous\_feature\_importances is not None:

...

חלק זה של הקוד משווה בין החשיבות הנוכחית של התכונות לבין אלו שהיו במודל הקודם, כדי לבדוק אם הייתה השפעה כלשהי לנתונים החדשים על המודל.

1. **הדמיית עץ ההחלטות**:

graph = graphviz.Source(dot\_data)

graph.render(os.path.join(graph\_path, f'decision\_tree\_{datetime.datetime.now().strftime("%Y%m%d\_%H%M%S")}.png'))

חלק זה בקוד אחראי על יצירת ויזואליזציה של עץ ההחלטות ושמירתה כקובץ תמונה.

**תוצאות צפויות:**

* **דירוג התמונות**: המודל צפוי לשפר את יכולתו לדרג תמונות לפי רמת הקושי שלהן עם הזמן, ככל שמתקבלים נתונים חדשים ונערכים תהליכי אימון נוספים.
* **יכולת למידה מתמשכת**: המודל משפר את עצמו תוך כדי, שכן הוא מבצע עדכונים בהתבסס על נתונים חדשים שנאספים בכל הרצה.
* **חשיבות התכונות**: המודל מראה אילו תכונות משפיעות יותר על ההחלטות, מה שיכול לסייע להבין את תהליך קבלת ההחלטות של המודל בצורה טובה יותר.

**סיכום:**

הקוד המוצג כאן מספק פתרון יעיל לבניית מודל עץ החלטות שמסוגל ללמוד ולהשתפר עם הזמן. המודל משתמש בנתוני משתמשים ומסווג תמונות לפי רמת הקושי שלהן. בעזרת הדמיות ותיעוד הלוגים, ניתן לעקוב אחרי תהליך האימון ולראות כיצד המודל משתפר לאורך זמן.