CITY NEWS BEAT AI PROJECT

GOAL

- To create a video recommendation system in order to provide a custom experience for City News Beat app users.

HOW TO USE

- Go to https://city-news-beat.herokuapp.com/

- Click in "AI Engine" tab

- Specify the data to use to train the model:
    - **Option 1:** Upload two csv files with the data. Please, make sure you upload the correct file in the proper box, otherwise the program will not be able to function properly. (see **input** for more instructions on this)
      Note: Only csv files are allowed. You must upload both files before clicking "Submit".
    - **Option 2:** Check "Use the data available in the database" option to use already existing data collected in the database.
      Note: This is the preferred option. In case both a file is uploaded and this option is selected, the model will be trained on the data available in the database.
- Click "Submit" to select the settings and features you would like to activate in order to train the model. After this step, an adjacent form will appear.

- Select a model (default value: logreg).

Available models: 'logreg', 'multilogreg', 'knn', 'mlp'.

Note: 'xgboost' model is also implemented in the ai_engine.py file, but because it takes a long time to train on the data, it is not included on the website.

- Select the features you want to activate.
  Available features: primary_category, sub_category, sub_sub_category, vid_user_watched_ratio, vid_user_selected_watch_ratio, vid_avg_time_watched_ratio, vid_avg_interaction_span_day.
  More explanation about each feature **here**.

Note: It is recommended that at least 4 to 5 features be selected. If none are selected, then the default settings would run the model using all the available features.

- Input the desired NumKFolds under the "NumKFolds" label. NumKFold value ranges from 2 to 10. Default value is 5. More about NumKFolds **here**.

- Check Accuracy: You can select to check the F1Scores for a specific user, or for n randomly selected users (nF1scores).

Note: If you choose to upload the data from csv files, then the check nF1Scores option will not be available.

- Choose the "checkF1Scores" option to display the accuracy of the trained model (default). More about checkF1Scores **here**.

  - Click here to learn more about how model accuracy is determined: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Note: If this option is selected, you would have to select the user for whom you would like to generate a list of videos. If no user is selected, the model will run on the 1st user that appears in the dropdown menu.

The "Show video titles" box will also appear once the checkF1Scores is selected. Check the "Show video titles" box to display not only the id of each video, but also its title (optional).

- Choose the "nUserF1Scores" option to output the average of model's F1 scores (optional). More about nUserF1Scores **here**.

Note: When this is set to True, it overrides the checkF1Scores and showVidTitles settings. It only prints performance metrics over the span of randomly selected users, not of any model outputs on single users.

Example:

```
The F1 score is a value between 0 and 1. The closer to 1, the better the score.
The F1 score represents a balanced view between model precision and recall.
Average F1 score: 0.8538

Normalized confusion matrix results below. Values are between 0 and 1.

True Positive: 0.5 (model is correct when it says you like video)
False Positive (model error): 0.08 (model says you like a video when you don't)
True Negative: 0.38 (model is correct when it says you don't like video)
False Negative (model error): 0.04 (model says you don't like video when you do)
```

- If this option is selected, input the desired fraction under the "nUserFraction" label. nUserFraction value ranges from 0 to 1 (1: meas the F1Scores will be performed on 100% of the sample). Default value is 0.66. If 0 is selected, the model will run on a random user. More about nUserFraction **here**.

Note: This is only available when the nUserF1Scores setting is selected.

- Click the "Run" button to train the model with the specified settings. See **output** for more details on what will be displayed when you run the model.

INPUT

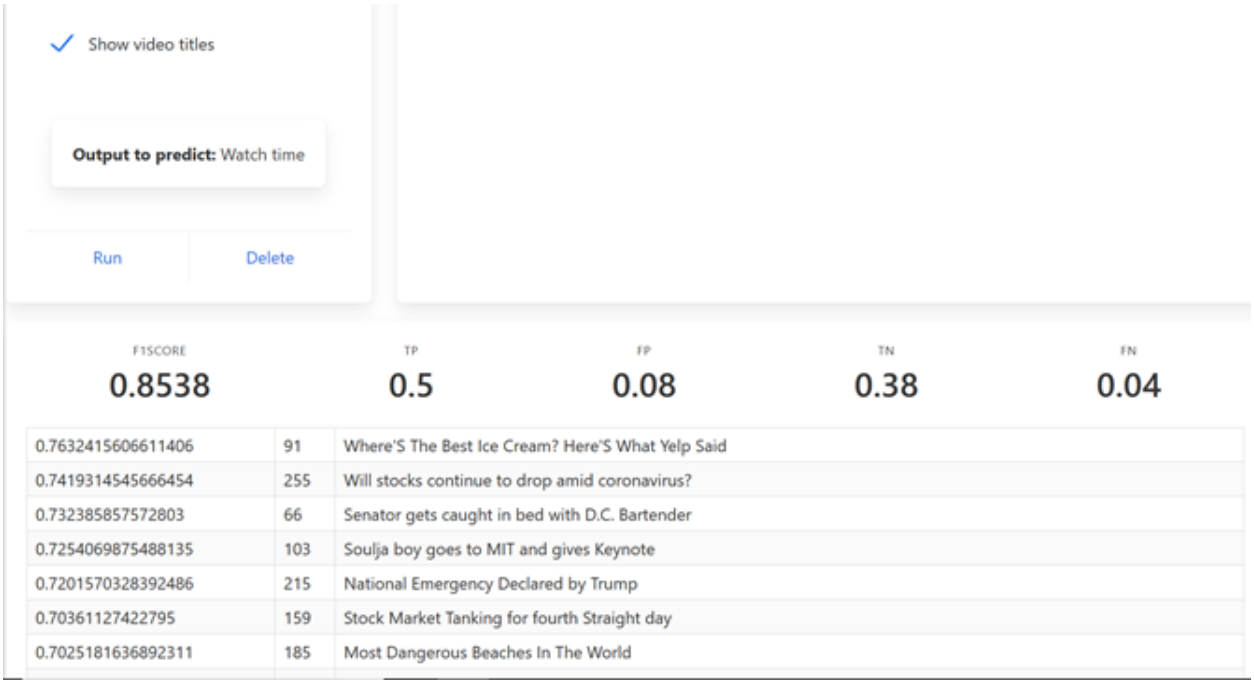
- CSV Files
  - Upload two CSV files for each of the Table screenshots shown below.
  - Make sure to match the formatting above with regard to column names, descriptions, and datatypes for each row in each of the files

UserInteractions Table:	**primary key = (uid, vid, date_watched)**	
column name	**description**	**datatype**
uid	user id of person that watched this video	string
vid	video id	string
date_watched	the date and time the user watched this video	string: yyyy-mm-dd hh:mm:ss
amount_of_time_watched	length of time the video was watched in seconds	int
vid_selected	whether or not the user specifically selected the video	boolean: 1 yes, 0 no
vid_skipped	whether or not the user specifically skipped the video	boolean: 1 yes, 0 no

VideoLibrary Table	**primary key = vid**	
column name	**description**	**datatype**
vid	unique video id	string
title	title of the video	string
primary_category	main category type video falls under	string: [sports, food, news, lifestyle]
sub_category	category type that stems from primary_category	string: [***sports***: football, bball, *any othe
sub_sub_category	category type that stems from sub_category	string: [for all sports subcats: highlights/
length	how long the video is in seconds	int
release_date	date and time the video was made available to the users	string: yyyy-mm-dd hh:mm:ss

OUTPUT

- A table will be displayed listing the videos the algorithm recommends for the selected user in ascending order based on the probability that the user will watch the video.
- The F1Scores, video ids, and Video titles will be displayed if selected in the settings.



BEHIND THE CODE

AI ENGINE README

General overview

The AI engine works by taking in the unique userID string as input, and two dictionaries for the settings. The model is run independently on each user. The videos that are used to train the model are the videos that the user has interacted with in any way (selected, skipped, watched for half the video time, etc), and the videos that are run through the "trained" model are the videos that the user has never interacted with. The output is a sorted array of tuples of those unwatched videos: first value in the tuple is the likelihood that the user likes the video, and the second value is the unique vid of that video. Output format as follows:

[(.987, 'vid1'),
(.845, 'vid2'),
...
(.476, 'vid553'),
(.123, 'vid554')]

For a binary interpretation of the output, a value above **0.5** means the user should like the video, and a value less than **0.5** means the user probably doesn't like the video.

The features the model uses for these video inputs are dependent on all of the user's interactions with those videos, as will be outlined in the feature descriptions below. The video categories are also taken into account as well. Using the correct model with the correct features is completely dependent on the data that is being used, so additional functionality is added to enable/disable model features, and to also select different models the engine will use. Testing functionality has also been added to show model performance over a single user or multiple users so that the correct model can be selected for the engine.

HOW TO SET UP DEPENDENCIES FOR AI ENGINE PRODUCTION

Step 1) Create the following tables in the PostgreSQL database with exact case sensitive table and column names, primary keys, and datatypes:

Table 1 Name:

userinfo

Column Name	Datatype	Description
uid	text	unique user id
dob	date	date of birth
index	integer	arbitrary index value

Primary Keys:

uid

Table 2 Name:

userinteractions

Column Name	Datatype	Description
uid	text	user id
vid	text	video id
date_watched	timestamp_without_time_zone	date user watched video
amount_of_time_watched	smallint	# seconds video watched
vid_selected	boolean	if user selected video
vid_skipped	boolean	if user skipped video
index	integer	arbitrary index value

Primary Keys:

uid, vid, date_watched

Table 3 Name:
videolibrary

Column Name	Datatype	Description
vid	text	unique video id
title	text	title of video
primary_category	text	category video is in
sub_category	text	second category video is in
Sub_sub_category	text	third category video is in
length	smallint	length of video (in seconds)
Release_date	time_stamp_without_time_zone	date the video was released
index	integer	arbitrary index value

Primary Keys:
vid

Step 2) Modify the following variables in the AI Engine to correctly connect to PostgreSQL

```
dbusername = 'username of database'
password = 'password'
host = '127.0.0.1'
port = '5432'
database = 'database name tables are in'
```

Note: If the database is hosted on Heroku, use the environment variable DATABASE_URL.

Step 3) The following Python libraries need to be in the environment for the script to run correctly (some are probably already in base Python):

```
scikitlearn
pandas
numpy
sqlalchemy
xgboost
random
warnings
```

PUTTING THE MODEL INTO PRODUCTION

The 3 variables to be modified by the user in order to correctly run the AI engine are settings, featureSettings, and uid. They are shown below in their default settings for how they should be set in production. The modelType setting can be set to any one of the model names that are provided in the dictionary of models in the script. They are as follows: **'logreg'**, **'multilogreg'**, **'knn'**, **'mlp'**, **'xgboost'**. Any number of featureSettings can be enabled/disabled as desired.

```
settings = {'modelType' : 'logreg',
            'checkF1Scores': False,
            'numKFolds': 5,
            'showVidTitles': False,
            'nUserF1Scores': False,
            'nUserFraction': 1
           }


featureSettings = {'primary_category': True,
                   'sub_category': True,
                   'sub_sub_category': True,
                   'vid_user_watched_ratio': True,
                   'vid_user_selected_watch_ratio': True,
                   'vid_avg_time_watched_ratio': True,
                   'vid_avg_interaction_span_days': True}

uid = 'user ID string'
```

SELECTING THE CORRECT MODEL/FEATURES AND TESTING PERFORMANCE

Model Settings

The other options in the settings dictionary are for testing the performance of the model you are running.

- **checkF1Scores:**
  Turn this option on to test the performance of the model on the single user you are running it on.

- **numKFolds:**
  Should be a value between 2 and 10. Default setting is 5. This tells the number of times the model should split the data up for testing and training when it is using cross validation for checking performance.

- **showVidTitles:**
  Turn this option on to include the video title in the tuple of the output

- **nUserF1Scores:**
  This is turned on to check model performance on a specified number of randomly selected users. This could potentially take a long time since it would be running the model on each of the selected users. This is good for getting a general sense of how the model is performing over a wide array of users, instead of a single user. When this is set to True, it overrides the checkF1Scores and showVidTitles settings. It only prints performance metrics over the span of randomly selected users, not of any model outputs on single users.

- **nUserFraction:**
  Value between 0 and 1 that sets the fraction of total users that will be used to check the model performance over that range of users when the nUserF1Scores setting is enabled.

Model Features

Enabling/Disabling certain features could have positive or negative impacts on the model you are running. The only way to really tell what will be useful for the data is to guess and check what works and what does not (the same applies to the model that is selected).
Descriptions of these features are below:

- **Primary_category:**
  primary_category for every video in the video library

- **sub_category:**
  sub_category for every video in the video library

- **sub_sub_category:**
  sub_sub_category for every video in the video library

- **vid_user_watched_ratio:**
  the number of distinct views divided by the total number of users for every video in the video library

- **Vid_user_selected_watch_ratio:**
  the number of times that a user has selected the video divided by the total number of users that have watched the video for every video in the video library

- **vid_avg_time_watched_ratio:**
  the average amount of time that users have spent watching the video for every video in the video library

- **vid_avg_interaction_span_days:**
  the average span of time that between a user watching the video and when that video was released for every video in the video library

TABLE OF CONTENTS

GOAL	1
HOW TO USE	1
INPUT	4
OUTPUT	5
BEHIND THE CODE	6
AI ENGINE README	6
GENERAL OVERVIEW	6
HOW TO SET UP DEPENDENCIES FOR AI ENGINE PRODUCTION	7
PUTTING THE MODEL INTO PRODUCTION	9
SELECTING THE CORRECT MODEL/FEATURES AND TESTING PERFORMANCE	9
MODEL SETTINGS	9
MODEL FEATURES	10