DOUGLASCOLLEGE

COMMERCE AND BUSINESS ADMINISTRATION
COURSE OUTLINE AND SCHEDULE

CSIS 4260-002: SPECIAL TOPICS IN DATA ANALYTICS

Final Exam, Wednesday, December 9, 2020

Instructions

- **Create a folder and rename it to include your name and course number (e.g., pAdams-CSIS4260-FinalExam).**
- **Create a sub-folder for each of the two questions to have chosen to do.**
- **All the files you are required to submit for each question should be placed inside that question's folder.**
- **Note, you will lose points if you just cut and paste materials from close exercises (e.g., If I see the same comments, variable names, etc. from class exercises being using in your code).**
- **If you just copy code from online sources instead of following what you are instructed to do, your work will be disqualified, and you will face further consequences.**
- **If in-class cheating is determined (i.e., you shared your work with another student in the class), your work will be disqualified, and you will face further consequences.**
- **Note: You are required to be logged into the exam session throughout the duration of the exam or until you submit your work.**

- **The exam consists of 3 questions. PLEASE DO ANY TWO QUESTIONS OF YOUR CHOISE.**

Question 1

You are provided a sample of dataset collected from Spotify Web API, which captures several music attributes. These attributes include audio features (https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/) and track features (https://developer.spotify.com/documentation/web-api/reference/tracks/get-track/).

Below I provide a description of the features you will select for analysis (the rest are defined in the Spotify links provided above). **The goal of the analysis is to predict the positiveness of a track**.

Feature	**Description**
duration_ms	The duration of the track in milliseconds
accousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

danceability	Danceability - how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
instrumentalness	Predicts whether a track contains no vocals (range 0.0 to 1.0). The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
loudness	The overall loudness of a track in decibels (dB). Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
speechiness	Speechiness detects the presence of spoken words in a track (range 0.0 to 1.0). The more exclusively speech-like the recording, the closer to 1.0 the attribute value.
tempo	The overall estimated tempo of a track in beats per minute (BPM). Tempo is the speed or pace of a given piece and derives directly from the average beat duration.
valence (outcome variable)	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

1. Import the data for analysis and select only the columns/features shown in the table above.
2. Convert duration (duration_ms) from milliseconds to minutes (there are 1000 milliseconds in a second and 60 seconds in a minute, so divide duration_ms by 60000) and call the new column “duration. Thereafter, drop the original duration_ms column (hint: a lambda expression can help here).
3. Record valence (outcome variable) into a dichotomous dummy variable and call the new column “positive”. Record valence such that if the value is 0.5 and greater, positive has a value of 1, otherwise positive has a value of 0. Thereafter, drop the original valence column.
4. Split the dataset into 65% training set and 35% test set.
5. Using the full dataset, fit a random forest classifier and obtain feature importances. Fit a random forest classifier with the following parameters: max_depth=5, n_estimators=10, max_features=1.
6. Create a dataframe using column names and feature importance scores and give the resultant dataframe the following column names: [“Feature”, “Importance”].
7. Print the dataframe with values sorted in a descending order using Importance column (hint: use the following syntax: df.sort_values(“col_name”, ascending=False).
8. In a markdown cell, write a paragraph of text explaining feature importances results and their implications about music positiveness (note: the width of the text should be the width of the screen – therefore wrap the text to the next line in the cell to avoid one long line of text).
9. Split the dataset into 65% training set and 35% test set.
10. Using the Pipeline class of scikit-learn, fit the following classification models: **Logistic Regression, RBF SVM (i.e., SVC with gamma coefficient), Decision Tree, Random Forest, MLPClassifier, and XGBoost**. Set the maximum number of iterations for the logistic regression to 5000 and that of MLP to 15000. Furthermore, for the MLP, set the learning rate to adaptive, solver to ‘lbfgs’ and alpha to 0.01. For the RBF SVM/SVC, set gamma coefficient to 2 and C coefficient to 1. For the random forest, use the same parameters you used in 5) above and for the decision tree, set max_depth parameter to 5.
11. Use the prediction accuracy score to select the best model.

12. You are provided a file called “new_data.csv” which contains a sample of 5 new data records, each of which includes track and other features. Valence (i.e., track positiveness) is not provided for the new data. The goal is to use the model you selected in 11) above to predict whether a given track in the new data is positive or not.
13. Import the data for analysis and select the features are you did before (again, valence is not provided). Convert duration_ms to minutes as before and rename the column to “duration” as before. Delete the original duration_ms variable.
14. Using the “best” model, predict whether a given track in the new data is positive or not.
15. Create a function called “display_predicted_class” and in it use a loop to display the predicted class for each new track. If the predicted class is 1, you should display “The predicted class for track x, is 1, which means that its music is positive”; otherwise, you should display “The predicted class for track x, is 0, which means that its music is negative”. Note: x=1,2,3,4,5, respectively.

Question 2

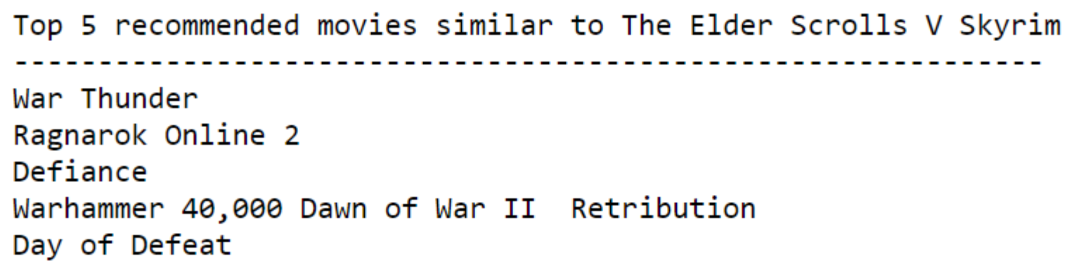
Background

You are provided with a sample dataset from Steam, a popular PC gaming hub. This dataset consists of user behaviors, with the following columns: user-id, game-title, behavior-name, and weight. The original dataset contained included 'own' and 'play' as user behaviors. I have filtered out “own” behavior to leave only “play” behavior. The weight column represents the number of hours a user has played the game. The goal of the analysis is to develop a game recommender system.

What you should do

1. You are provided two files: one called “game_titles_data.csv” and the other “video_games_data.csv”. Import the data for analysis and drop the behavior column from the dataframe for the video_games_data file..
2. Using the dataframe for the video_games_data file, pivot the data for analysis. **Challenge**: because of repeating values in the index column (i.e., Game_Title), the pivot() function will return error. **Work-around**: instead, using the following work-around to pivot the data: *df_final = (df.set_index(['Game_Title','User_ID']).unstack()).fillna(0)*
3. If you run df_final.head(), you should see the data is in appropriate “pivot” form.
4. Define a NearestNeighbors() model with the following parameters: metric='cosine', algorithm='auto', n_neighbors=20, n_jobs=-1)
5. Fit the model you defined in 4) above to the pivoted data (i.e., df_final)
6. Extract the distances and game title ids from the model.
7. The goal is the use the model to provide recommendations. Please use the results from your model to provide game recommendations to users. Create any necessary dataframes to aid the analysis.
8. Define a helper value-returning function called “*get_id_from_title*()” that takes one parameter called “term”. This function should return the ids of game titles related to the search term. You can utilize the dataframe for game_titles_data here.

9. Define a function called "*make_recommendation*" that takes <u>two</u> parameters, one called "num" (i.e., the number of recommended games to show to the user) and the other called "game_title" (i.e., user's search term). In a typical user environment, the "game_title" parameter would be a user's search query. Given the "game_title" argument when your function is called, you should display 5 (i.e., num=5) top recommended games like the game title the user entered. For example, if the user's search term is the first game title (i.e., "The Elder Scrolls V Skyrim"), your make_recommendation should display the following message:



```
Top 5 recommended movies similar to The Elder Scrolls V Skyrim
-------------------------------------------------------------
War Thunder
Ragnarok Online 2
Defiance
Warhammer 40,000 Dawn of War II  Retribution
Day of Defeat
```

Question 3

Background

Osteosarcoma is the most common type of bone cancer that occurs in adolescents in the age of 10 to 14 years. You are provided with a dataset that is composed of Hematoxylin and eosin (H&E) stained osteosarcoma histology images, collected by a team of clinical scientists at University of Texas Southwestern Medical Center, Dallas. Archival samples for 50 patients treated at Children' s Medical Center, Dallas, between 1995 and 2015, were used to create this dataset. Four patients (out of 50) were selected by pathologists based on diversity of tumor specimens after surgical resection. The dataset you are provided is a sub-sample of 547 images labeled as follows: "non-tumor", "non-viable-tumor", "viable", and "viable:non-viable". The goal of the analysis is to build image classification model and use it to classify new images.

What you should do

1. The images are in a folder called "Osteosarcoma_dataset1" and the labels are in a folder called "image_labels". Import/read the data for analysis.
2. Create a dataframe using the image labels file and do value count of the "Label" column. You will notice that "viable:non-viable" category has only 13 values. We will not use this category in analysis.
3. Using ***cat.remove_categories()*** method or other means (hint: "isin() is another method used in class examples to return dataframe with only desired categories), remove rows/records containing the "viable:non-viable" category.
4. <u>**Note**</u>: If you use "**cat.remove_categories()**, it sets unused categories values to np.nan in the data. Therefore, you need to drop rows with missing values again after the previous step.
5. Using the images column of the filtered labels dataframe as the list of images to process, read each image to an array and create an array of images (hint: if you did this correctly, the resultant array of images should have the shape (534, 256, 256, 3).

6. Extract the “Labels” column from the filtered labels dataframe and store it in a variable.
7. Using LabelEncoder and its fit_transform() method, transform the labels into indices, storing the result in another variable (e.g., “label_index”).
8. Randomize images array and label_index variable concurrently using shuffle() function, setting random_state=42 and store result as final_df.
9. Extract X (i.e., final_df[0]) and y (i.e., final_df[1]) vectors, respectively and thereafter split the data into 75% training set and 25% test set.
10. Scale the “X” samples into [0,1] range by dividing each by 255.
11. Convert “y” samples to categorical dummies using np_utils.to_categorical() method (note: there are 3 classes).
12. Define a sequential convolutional neural network (CNN) model with 3 convolutional layers of 30, 15, and 60 units, respectively. Set the kernel_size of each layer to (3,3) and pool_size to (2,2).
13. Add regularization dropout of 0.5.
14. Add 2 dense fully connected output layers with 64 and 32 units, respectively, specifying “relu” as the activation function for each.
15. Add a dense classifier layer, specifying “softmax” as the activation function.
16. Compile the model specifying categorical entropy as loss function, Adam optimizer, and accuracy metric. Print a summary of the model architecture.
17. Train/fit the model using a batch size of 128 and 30 epochs.
18. After you obtain the results, plot the training history.
19. The folder called “new_sample” contains 3 images, one for each of the classes/labels (image 0 is for “non-viable-tumor”, image 1 is for “non-tumor” and image 3 is for “viable” tumor). The goal is to use your model for out-sample-predictions (in this case, predict image class and see if the model replicates/reproduces “ground truth” (i.e., what we know about these images)).
20. Import that images for analysis and process them as you did the X samples above (i.e., create a matrix of images and then scale the result by dividing by 255).
21. Predict the class and probability of belonging to that class for each new image.
22. Create two helper functions, one called “print_predicted_class” and the other called “print_predicted_probability”. Using a loop and conditional logic to display the appropriate message (if you don’t want to use a loop, you can also predictions one image at a time – but that is not efficient!). For predicted class, print the following messages: if the predicted class is 0, display “The class of the image is [0], which means the image is a non tumor”. If the predicted class is 1, display “The class of the image is [1], which means the image is a non-viable tumor”. If the predicted class is 2, display “The class of the image is [2], which means the image is a viable tumor”. For the predicted probabilities, just display the class name and the predicted probability in the format “class name” [predicted probability].
23. Run your helper functions to display the results.

******** END OF EXAM *******