**HOW TO RUN SLEEP STAGE CLASSIFIER**

**AND SUBJECT CLUSTERING**

1. **Dependencies**

* MNE library – follow instructions at link below to install MNE developer version <https://martinos.org/mne/dev/contributing.html#installing-developer-version>
* Other libraries:
  + numpy
  + pandas
  + scipy
  + os
  + sys
  + glob
  + math
  + tqdm
  + pathlib
  + matplotlib
  + keras
  + sklearn
  + pickle
* Conda .yml files can be found in the ‘environments’ directory.

1. **Importing EDF files and annotations from SHHS database**

* Download from SHHS website
  + Go to the [SHHS database website](https://sleepdata.org/datasets/shhs/files) and sign in to enable downloading files.
  + Run nsrr download shhs/polysomnography/edfs/shhs1 to download the EDF files.
  + Run nsrr download shhs/polysomnography/ annotations-events-nsrr/shhs1 to download the accompanying annotation files.
* Importing in Python
  + Import the module shhs/polysomnography/polysomnography\_reader and call the function load\_shhs\_raw\_annotated\_edfs(edf\_path, annotations\_path) to load MNE Epoch objects for each EDF file present in the target directory edf\_path that has a corresponding annotation file in annotations\_path.

1. **Sleep\_stage\_classifier**

* All codes run within Sleep\_stage\_classifier directory
* Step 1: run Prepare\_model\_data.py
  + - Specify raw data location shhs\_base\_dir (Figure 1). It should have 2 subdirectories: edfs and annotations-events-nsrr. Within each subfolder, there is a folder shhs1 which stores the edf file and annotation file respectively (Figure 2)
    - Specify the output location (Figure 1)
    - The outputs are train/valid/test X and Y dataset saved in numpy npy format
      * X\_train.npy/Y\_train.npy
      * X\_valid.npy/Y\_valid.npy
      * X\_test.npy/Y\_test.npy

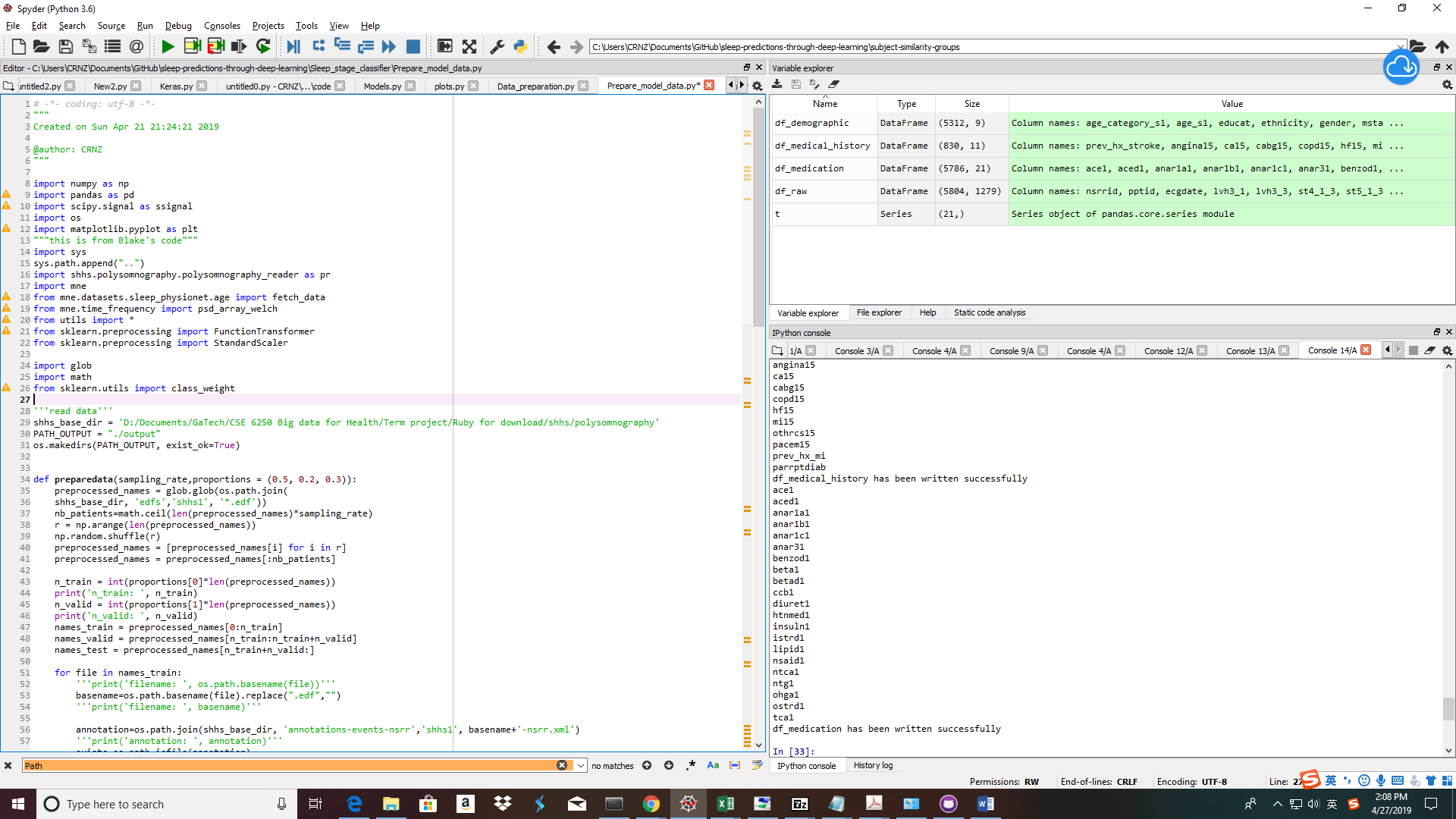


Figure 1: input required for Prepare\_model\_data.py

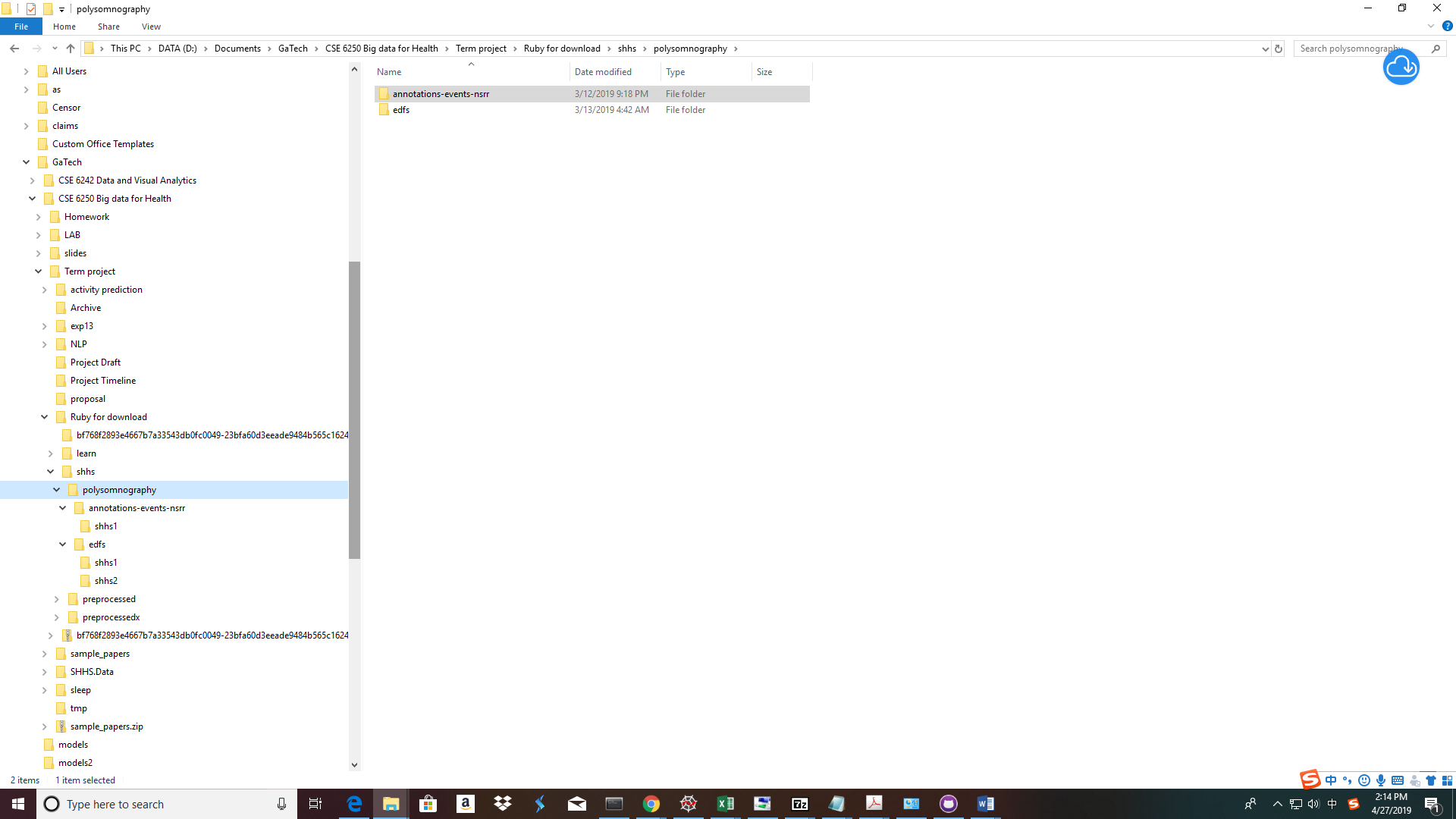


Figure 2: Expected folder structure of edf files and annotation files

* + Step 2: run run\_model.py
    - Need specify the model output folder
    - Also need specify the PATH\_TRAIN\_FILE, PATH\_VALID\_FILE and PATH\_TEST\_FILE, which should be the same the output location as specified in step 1

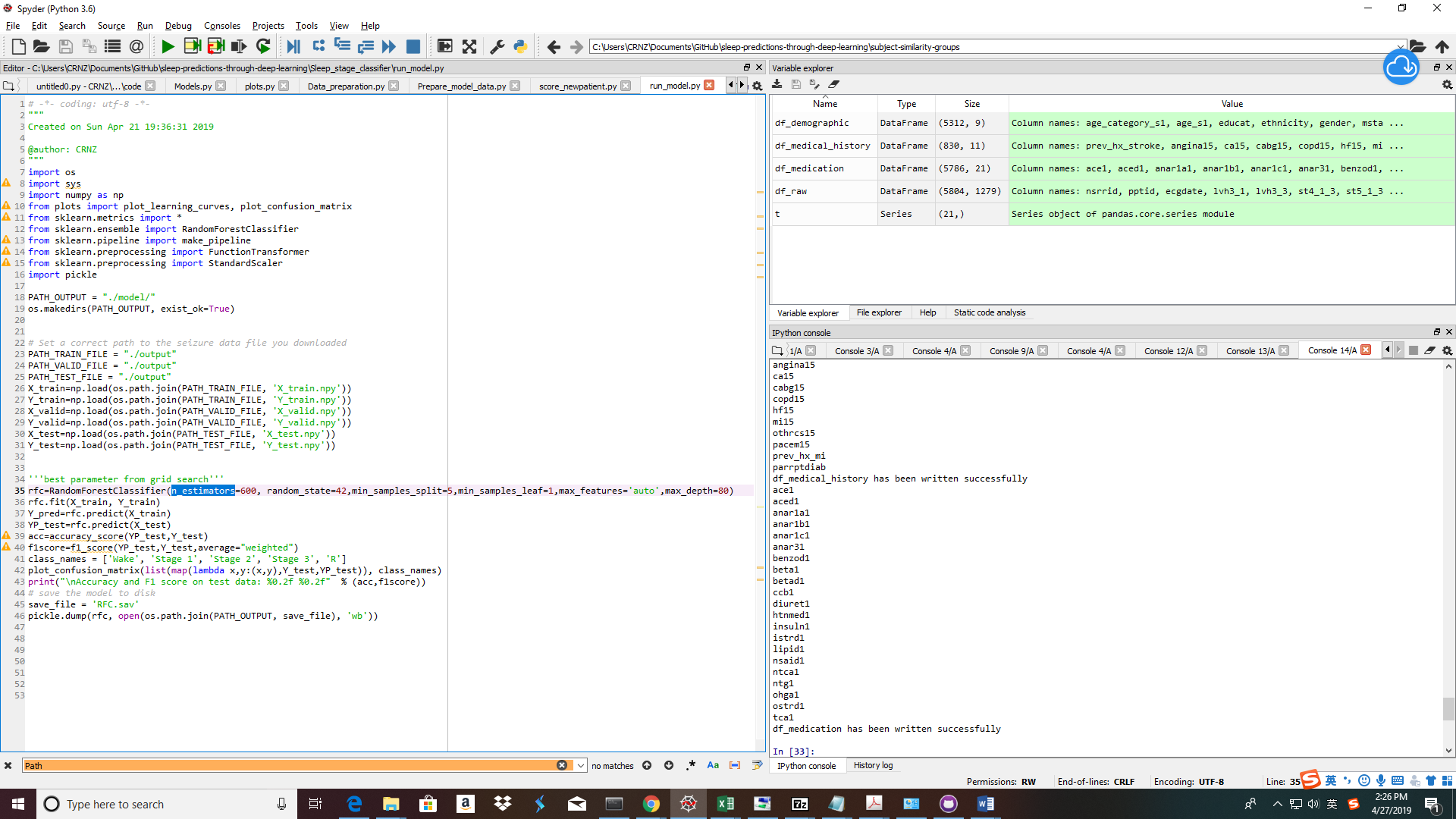


Figure 3: Required inputs for run\_model.py

* + - The output is a saved random forest model in .SAV format
    - It will also print the F1score and accuracy of the test model results. A confusion matrix is also saved
  + Step 3: score\_newpatient.py
    - Need specify the location of edffile
    - The location of X\_train: this is to standardize the scored file
    - The location of model
    - The location of the output folder for scored results
    - The final output is a csv file which has sleep stage for one single patient

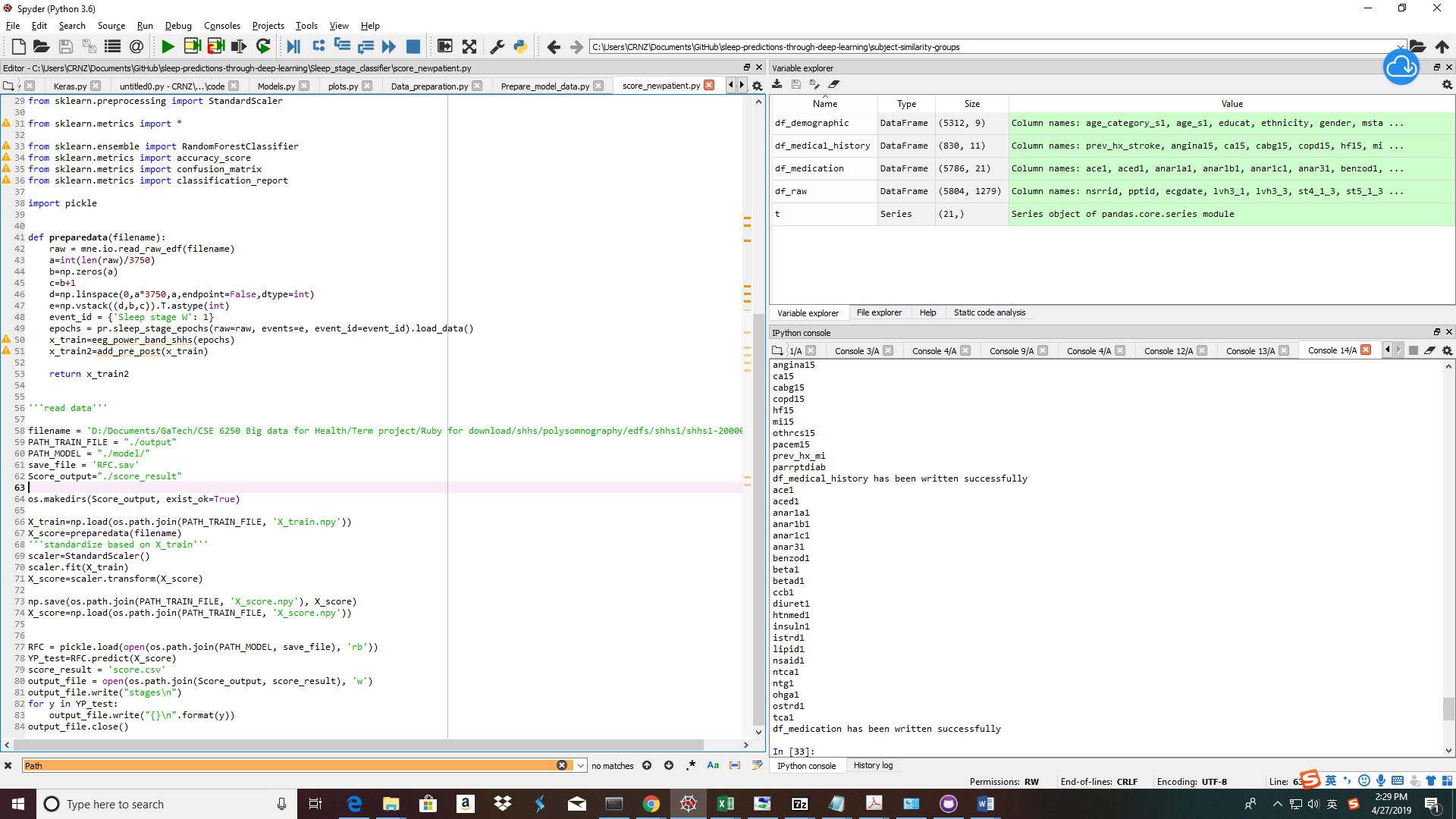


Figure 4: Required inputs for score\_newpatient.py

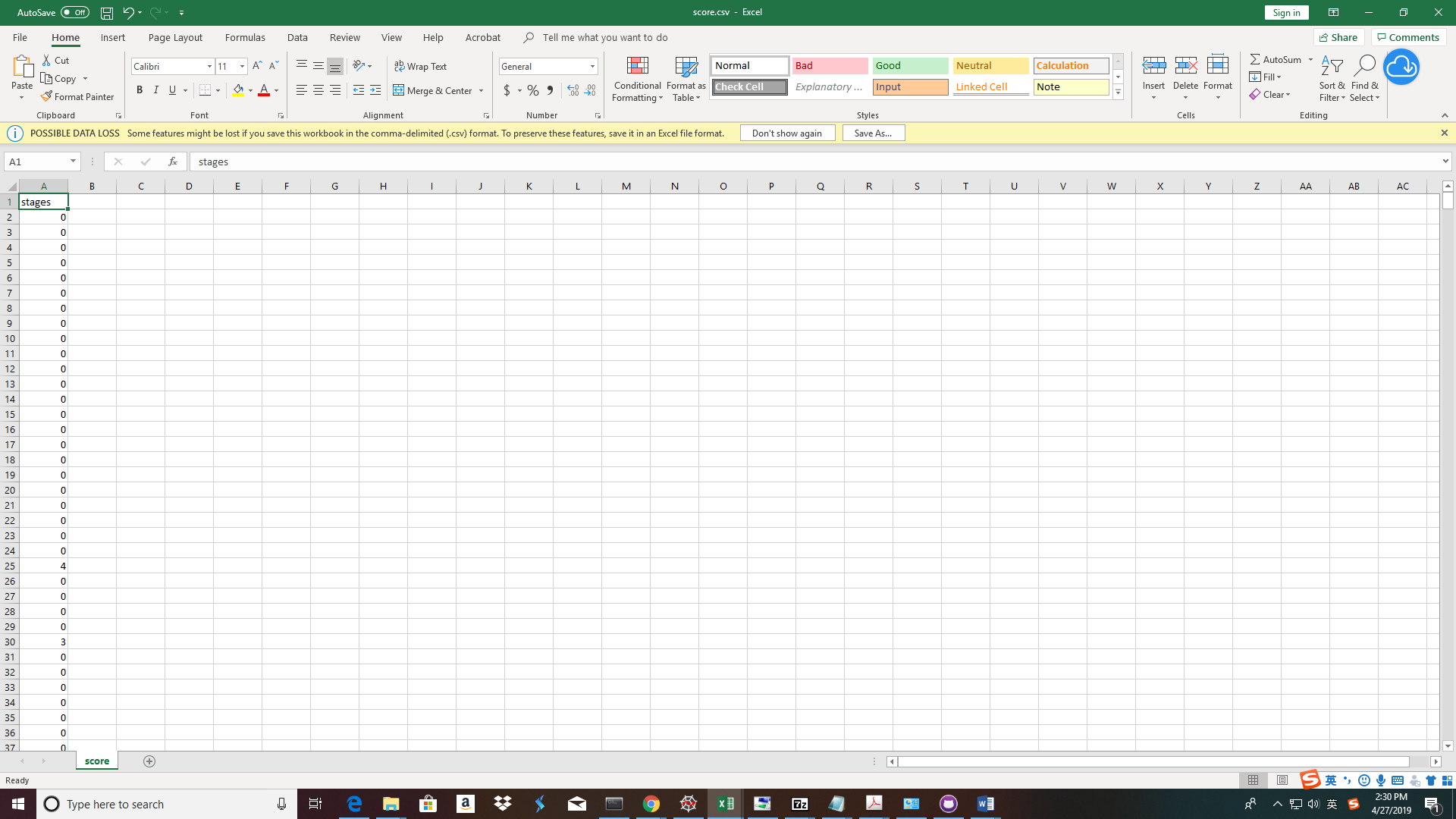


Figure 5: A snapshot of output:score.csv

1. **CNN\_CNN\_Model**
   * This is an alternative model built using CNN to extract features and feed into the next CNN model. The model spec is borrowed from (<https://towardsdatascience.com/sleep-stage-classification-from-single-channel-eeg-using-convolutional-neural-networks-5c710d92d38e>) but the data preparation pipeline used our own and uses PSD instead of raw EEG data
   * All codes run within CNN\_CNN\_Model
   * Step 1: run Data\_preparation.py
     + Specify raw data location shhs\_base\_dir. The folder structure within base follows the ones described in Figure 2: It should have 2 subdirectories: edfs and annotations-events-nsrr. Within each subfolder, there is a folder shhs1 which stores the edf file and annotation file respectively
     + Specify the output location
     + The % of data used for model training can also be modified
     + The output is saved CNN model in h5 format

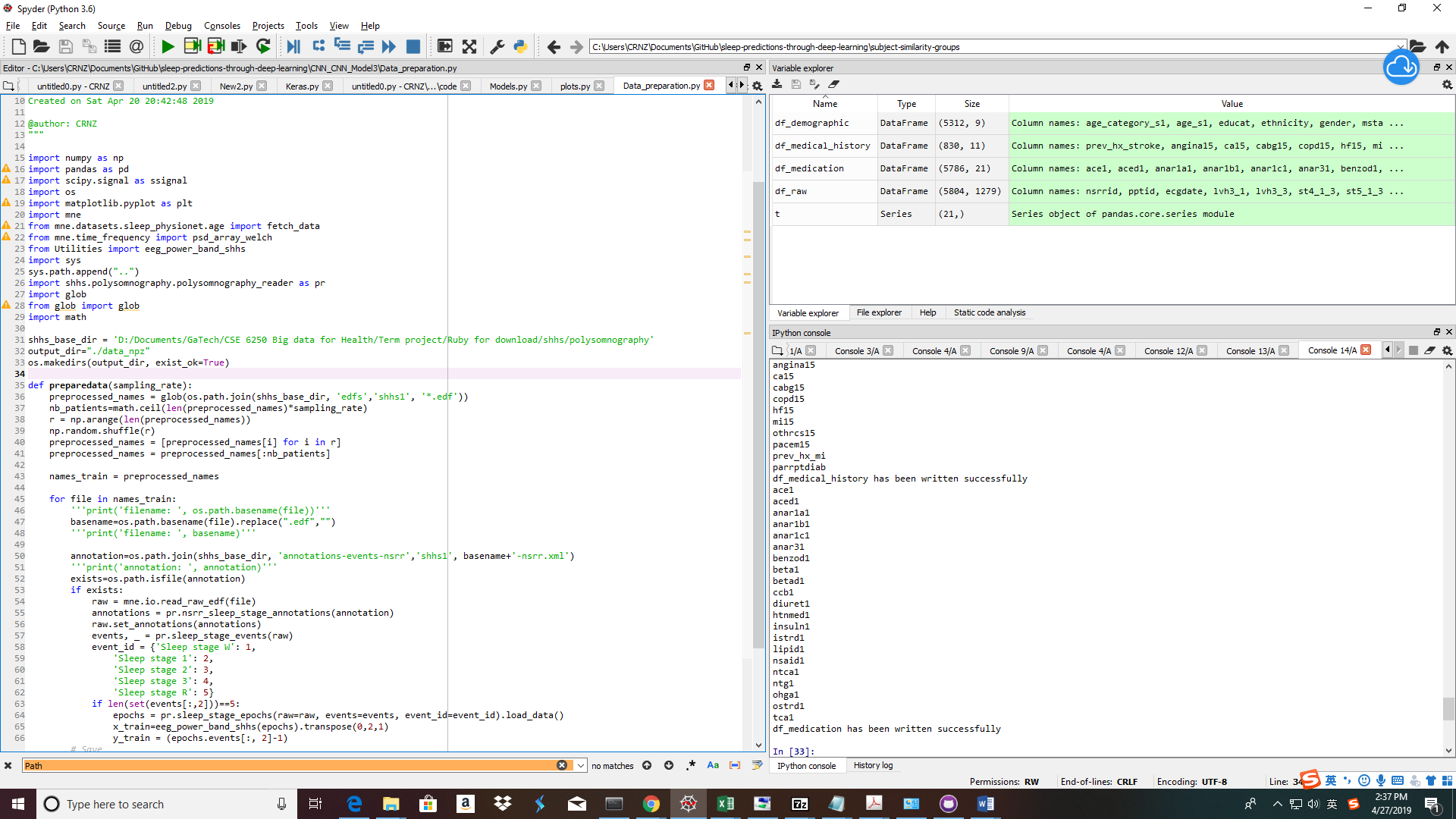


Figure 6: input required for Data\_preparation.py

* + Step 2: run Train\_Model.py
    - Data\_path: Specify the location of npz files as produced in Step1
    - File\_path and epochs: Specify the file name for the model and # of epochs
    - The output is saved CNN model in h5 format in the current directory, print F1score and accuracy in the screen and save the confusion matrix

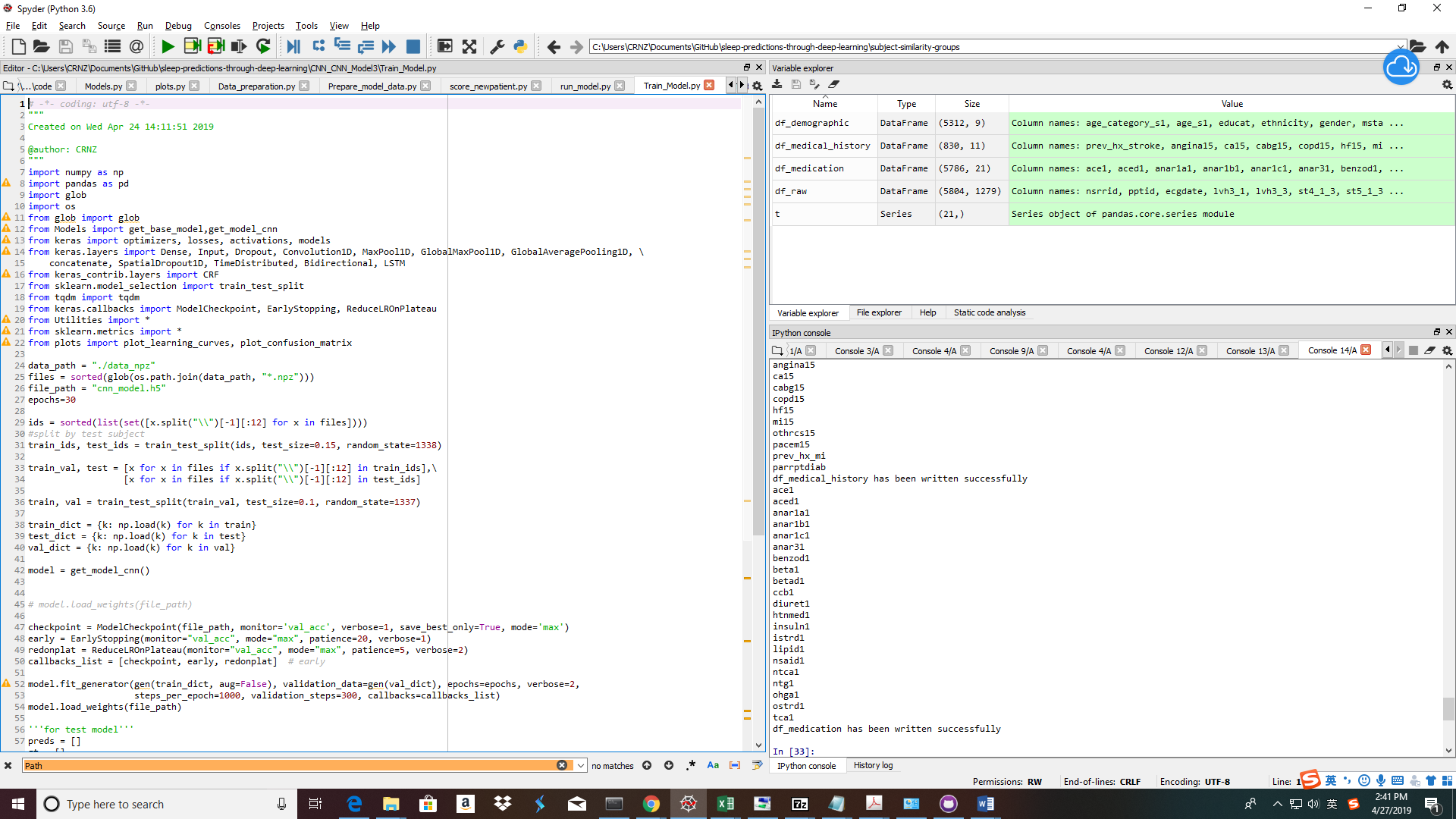


Figure 7: input required for Train\_Model.py

* + Step 3: Test\_Trained\_Model.py
    - This is a standalone program to just run the trained model on test set since Train.Model.py runs considerable amount of time
    - Need specify the input location of saved npz files, the saved model file name
    - The output is a saved confusion matrix graph and f1score and accuracy printed on the screen

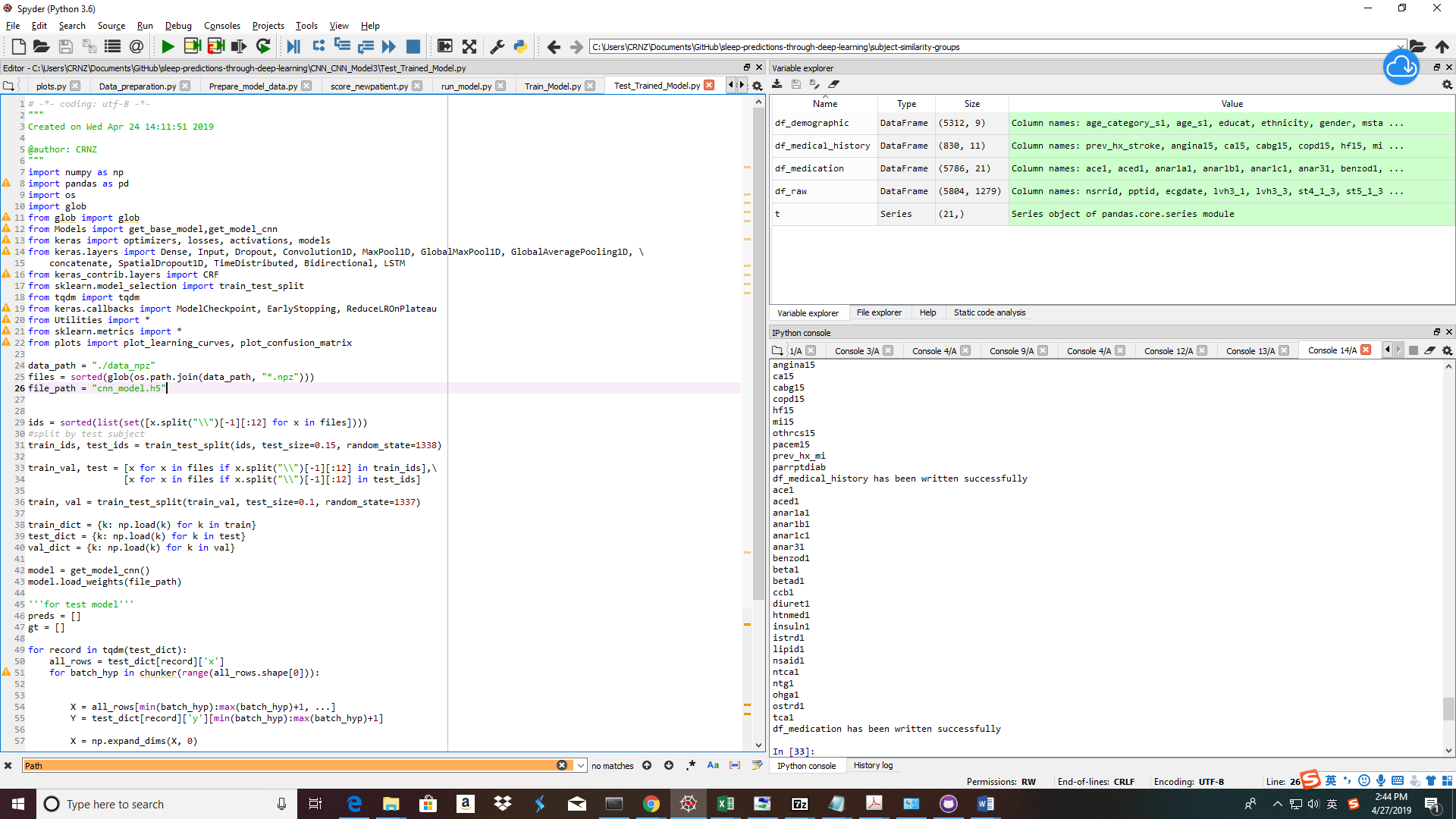


Figure 8: input required for Test\_Trained\_Model.py

* + Step 3: Scoring\_newpatient
    - Need specify the location of new patient’s edffile
    - The name of the model
    - The location of the output folder for scored results
    - The final output is a csv file which has sleep stage for one single patient

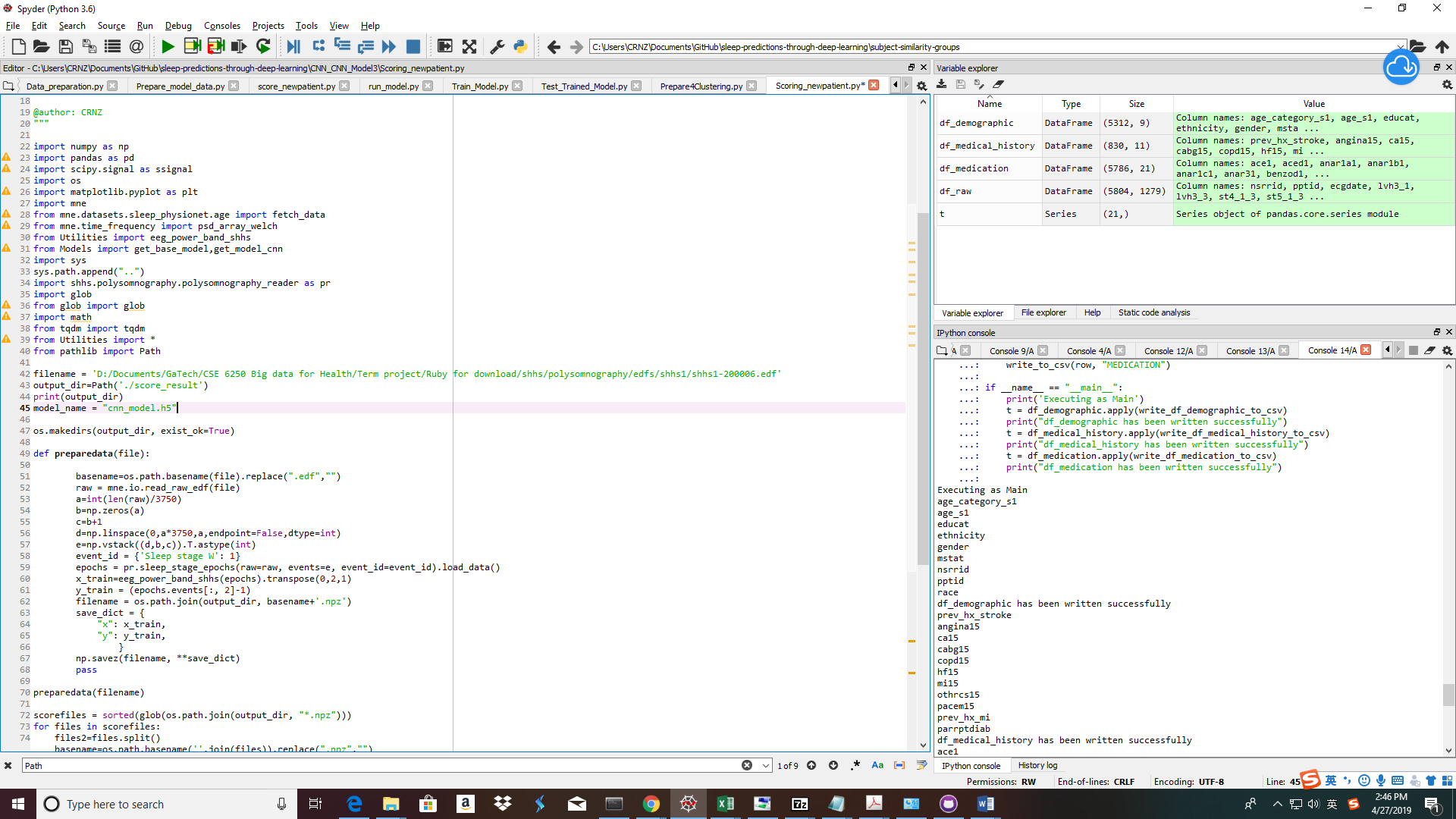
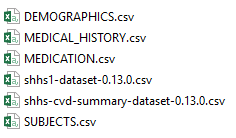


Figure 9: input required for Scoring\_newpatient.py

1. **Subject Similarity and Clustering**
   * Process raw SHHS data
     + Download shhs1-dataset-0.13.0.csv from the SHHS dataset and save in ‘subject-similarity-groups/data’.
     + ‘python reworkSubjectData.py’
     + This generates csv files for scala graphing.
       - 
   * Perform scala subject graphing
     + ‘docker start bigbox’
     + ‘docker attach bigbox’
     + ‘/scripts/start-services.sh’
     + ‘cd /mnt/host/{PATH\_TO\_SOURCE\_CODE}’
     + Update ‘host\_dir’ variable in ‘main.scala’
     + ‘sbt console’
     + Run code from ‘main.scala’
       - Output will be hdfs file ’SubjectSimilarities.scala.csv’
       - Copy this to ‘subject-similarity-groups /output’
   * Perform subject clustering with scikit-learn
     + ‘python Clustering.py’
       - This will generate ‘subject-similarity-groups /output/ClusterID.csv’
   * Find CVD risk predictions
     + Download shhs-cvd-summary-dataset-0.13.0.csv from the SHHS dataset and save in ‘subject-similarity-groups/data’.
     + Ensure ‘subject-similarity-groups/output/ClusterSimilarities.csv’ exists that was created by previous model steps.
     + ‘python OutputPerCluster.py’
       - This will print the risk factor prediction percentages as well as generate ‘subject-similarity-groups/output/RiskPrediction.csv’
     + ‘subject-similarity-groups/output’:
     + 