* + - * 1. **HOW TO RUN SLEEP STAGE CLASSIFIER**

Dependencies:

1. MNE library – follow instructions at link below to install MNE developer version <https://martinos.org/mne/dev/contributing.html#installing-developer-version>
2. Other libraries:
   1. numpy
   2. pandas
   3. scipy
   4. os
   5. sys
   6. glob
   7. math
   8. tqdm
   9. pathlib
   10. matplotlib
   11. keras
   12. sklearn
   13. pickle

Run the classifier

**For TA: See step 1.c for testing the model**

1. Sleep\_stage\_classifier
   1. All codes run within Sleep\_stage\_classifier directory
   2. Step 1: run Prepare\_model\_data.py
      1. 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)
      2. Specify the output location (Figure 1)
      3. The outputs are train/valid/test X and Y dataset saved in numpy npy format
         1. X\_train.npy/Y\_train.npy
         2. X\_valid.npy/Y\_valid.npy
         3. X\_test.npy/Y\_test.npy

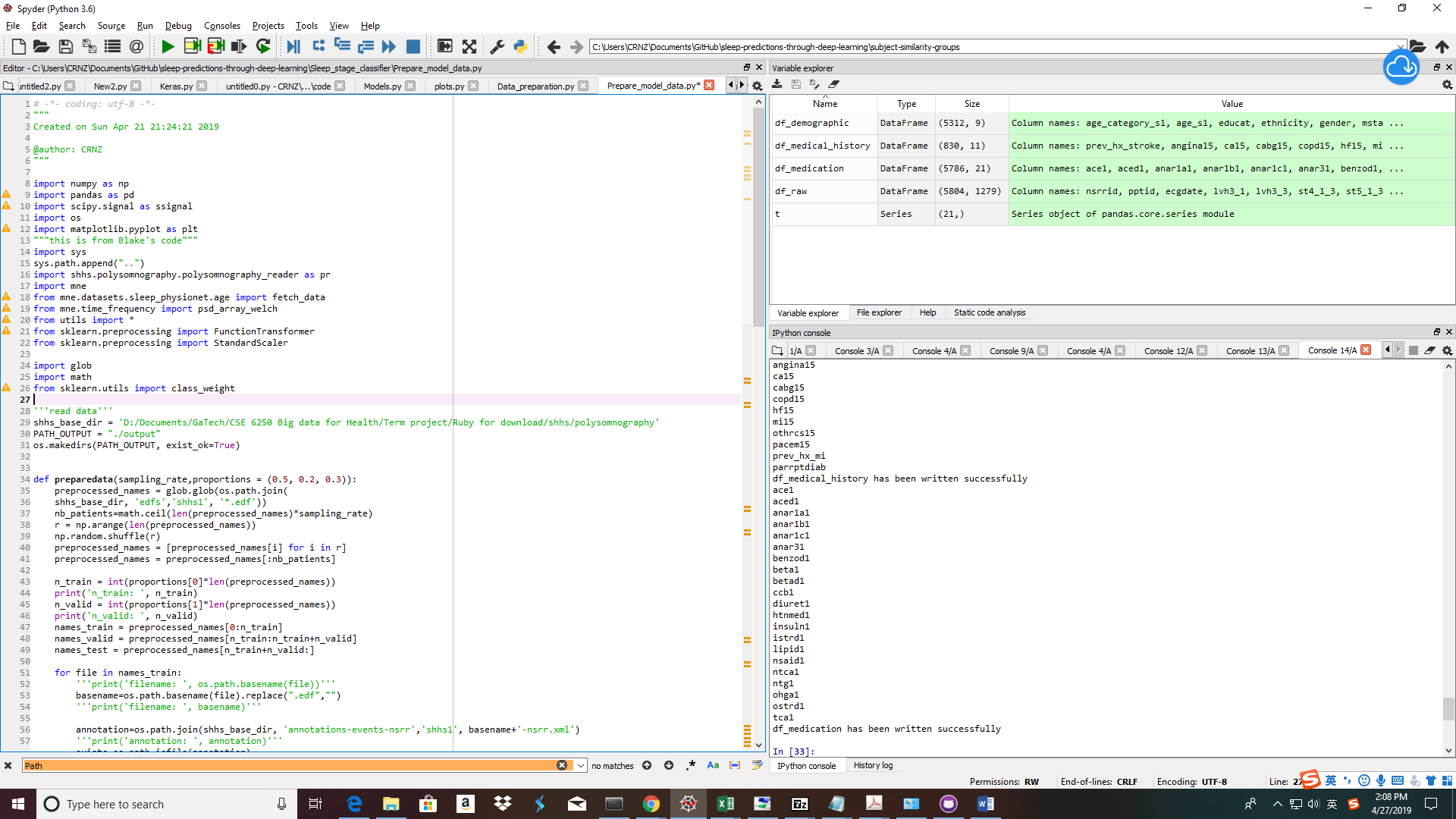


Figure 1: input required for Prepare\_model\_data.py

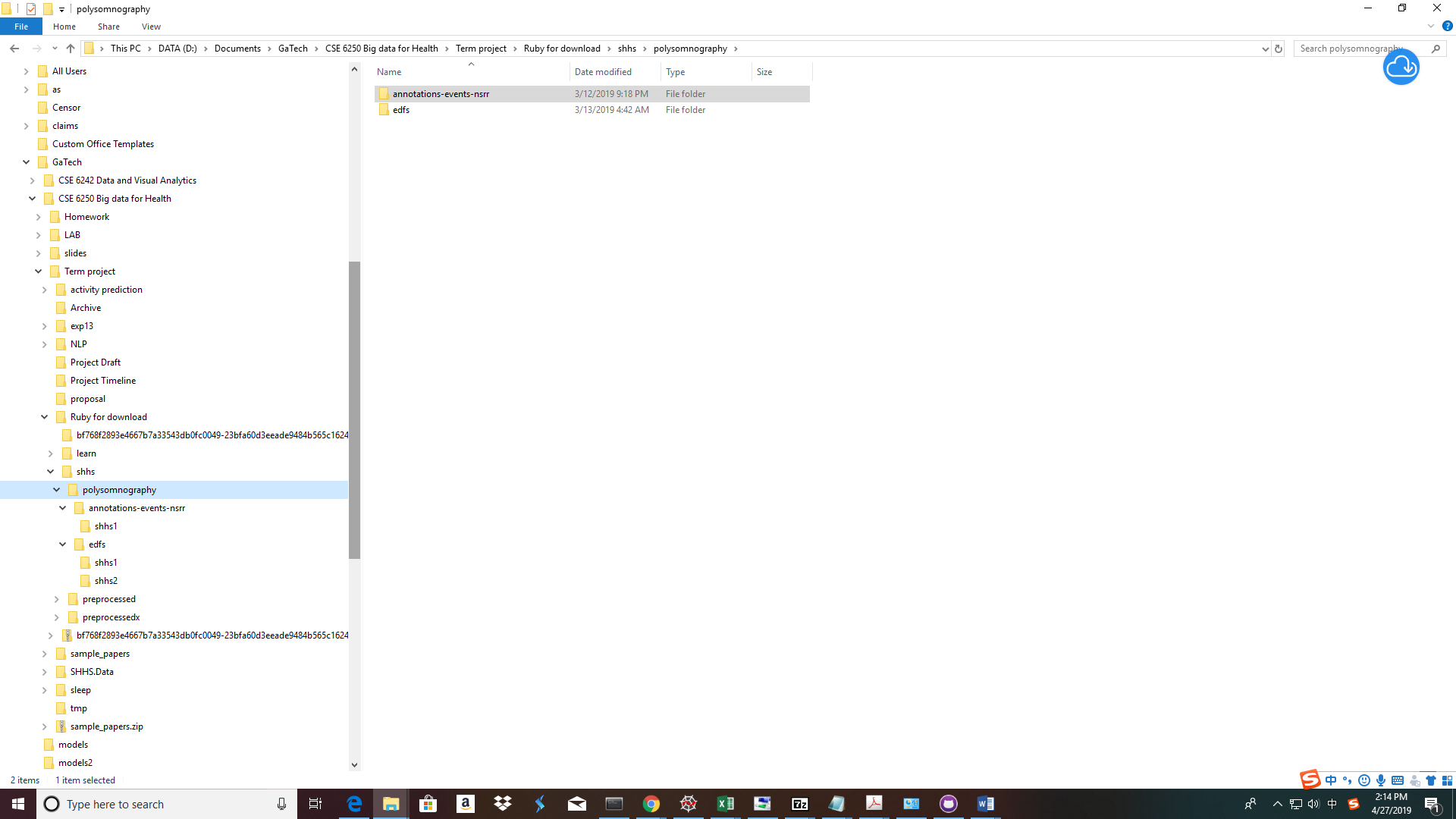


Figure 2: Expected folder structure of edf files and annotation files

* 1. Step 2: run run\_model.py
     1. Need specify the model output folder
     2. 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

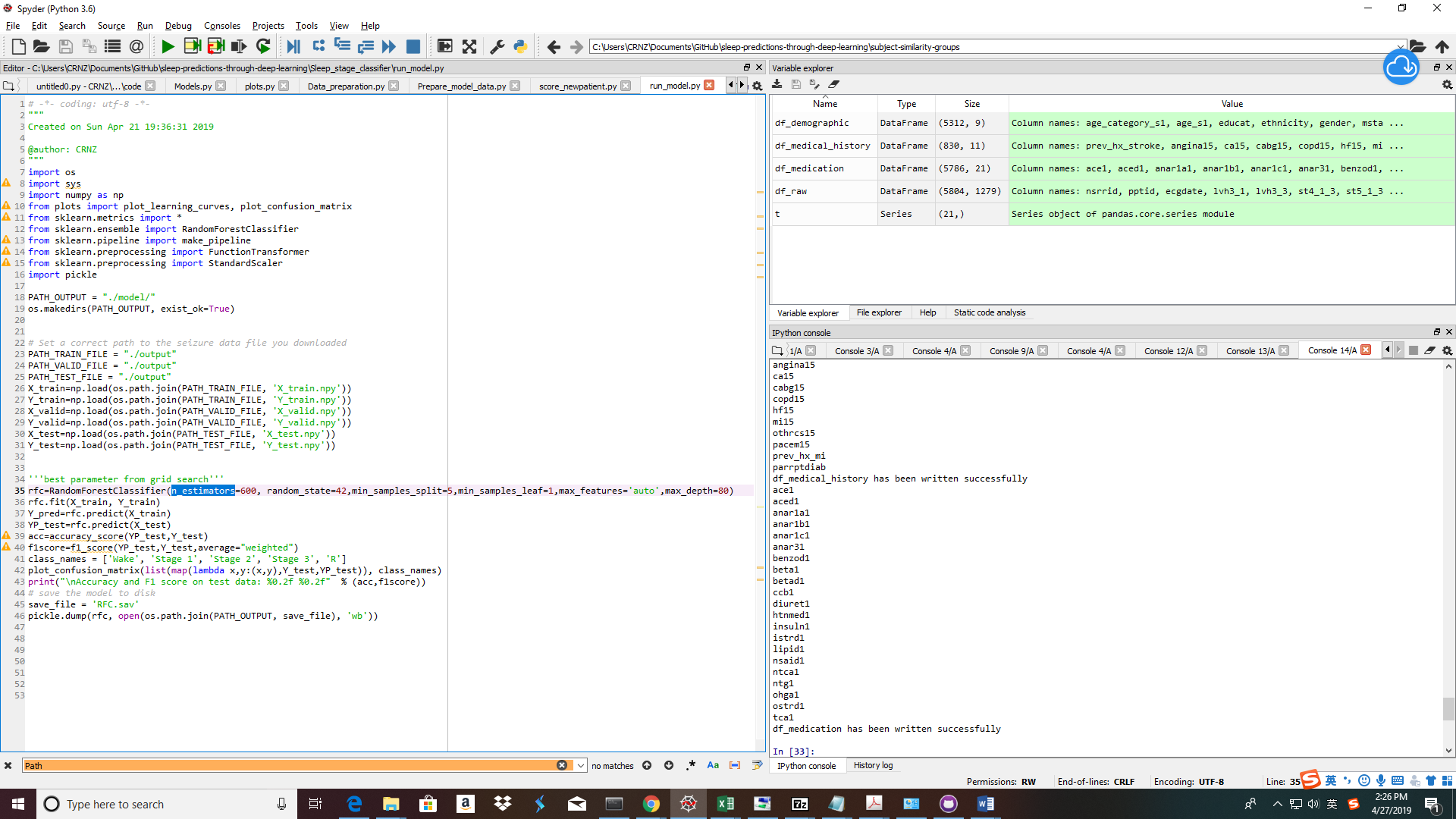


Figure 3: Required inputs for run\_model.py

* + 1. The output is a saved random forest model in .SAV format
    2. It will also print the F1score and accuracy of the test model results. A confusion matrix is also saved
  1. Step 3: score\_newpatient.py
     1. Need specify the location of edffile
     2. The location of X\_train: this is to standardize the scored file
     3. The location of model
     4. The location of the output folder for scored results
     5. The final output is a csv file which has sleep stage for one single patient

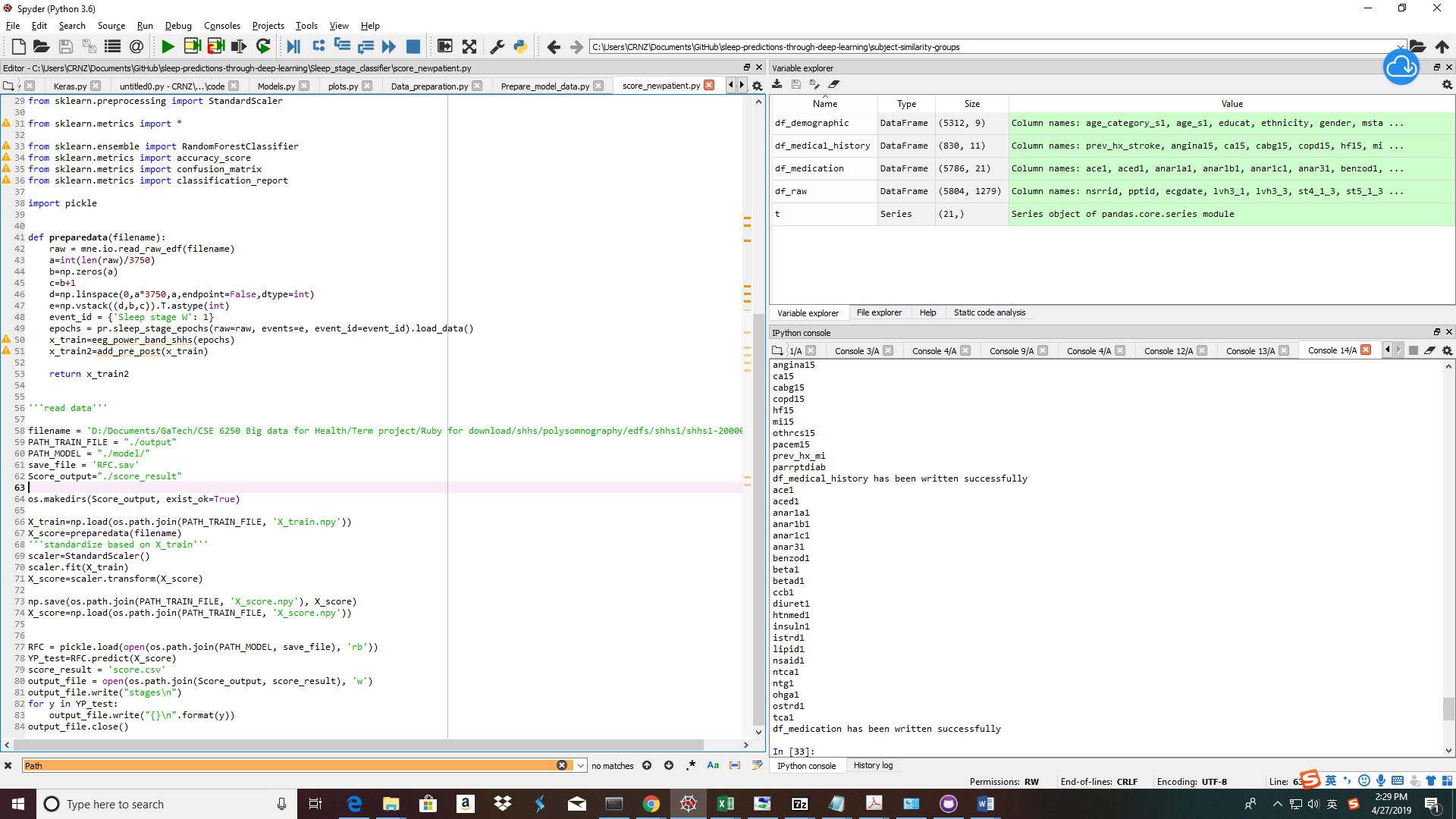


Figure 4: Required inputs for score\_newpatient.py

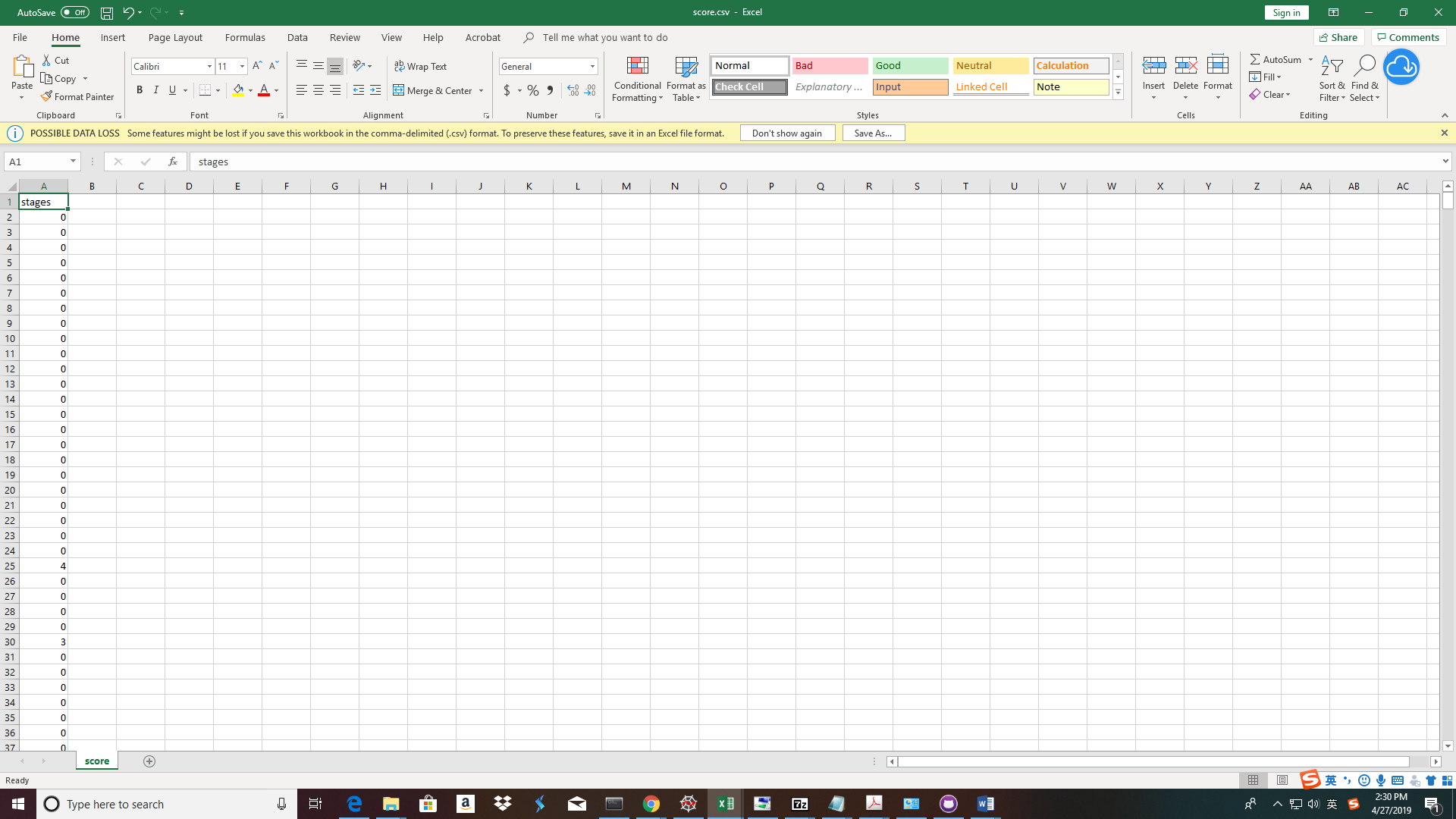


Figure 5: A snapshot of output:score.csv

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

* 1. All codes run within CNN\_CNN\_Model
  2. Step 1: run Data\_preparation.py
     1. 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
     2. Specify the output location
     3. The % of data used for model training can also be modified
     4. The output is saved CNN model in h5 format

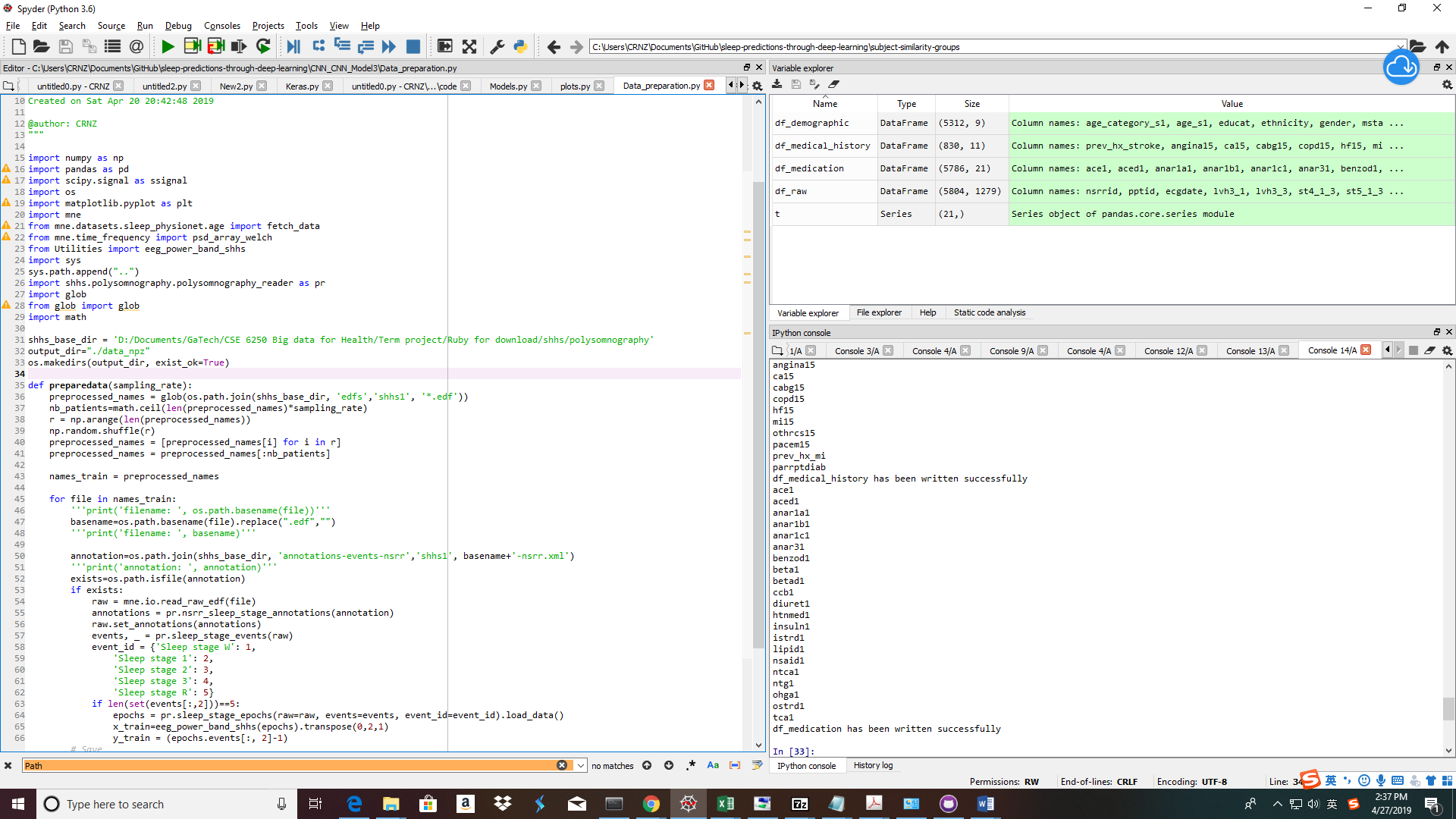


Figure 6: input required for Data\_preparation.py

* 1. Step 2: run Train\_Model.py
     1. Data\_path: Specify the location of npz files as produced in Step1
     2. File\_path and epochs: Specify the file name for the model and # of epochs
     3. 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

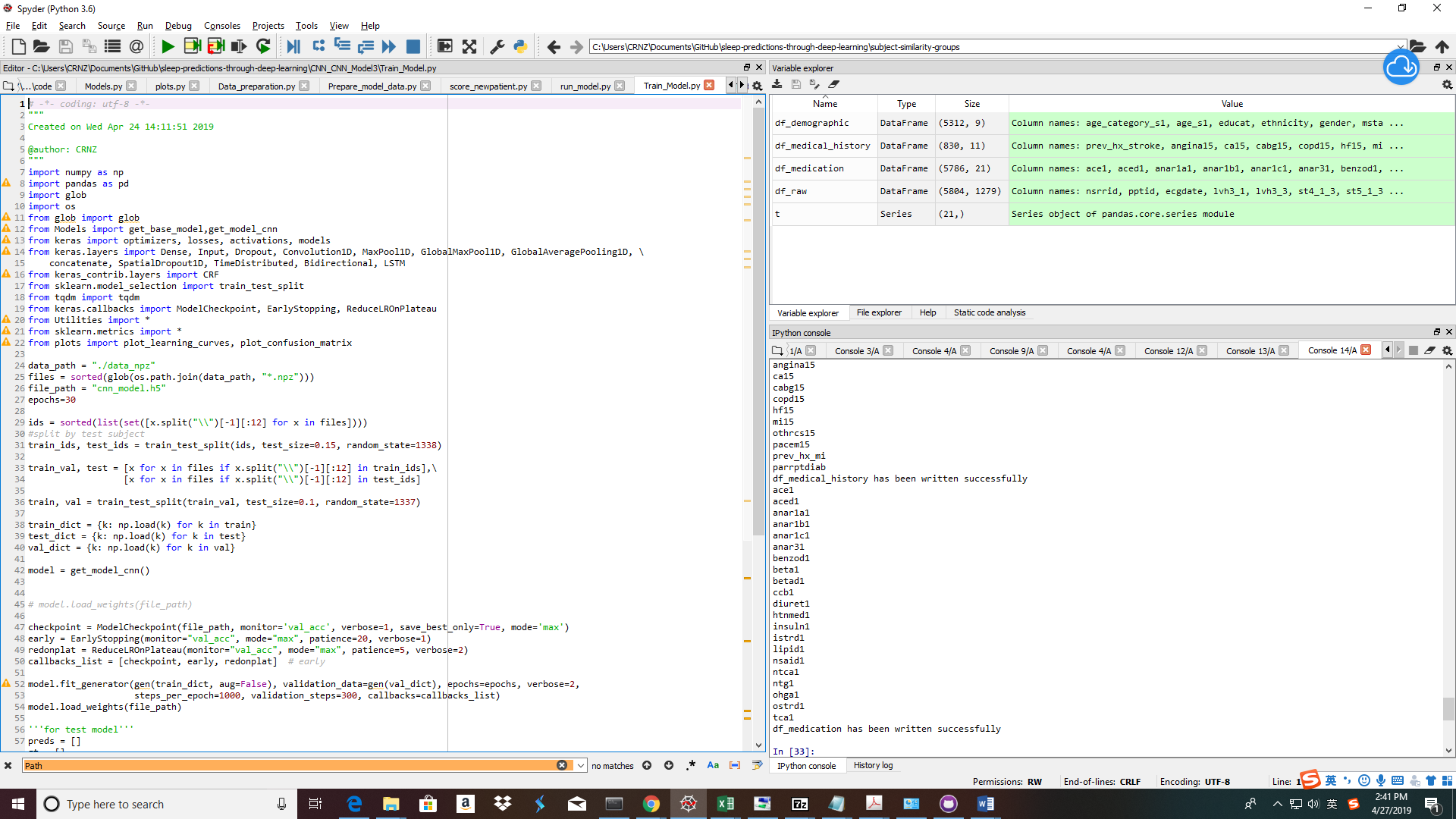


Figure 7: input required for Train\_Model.py

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

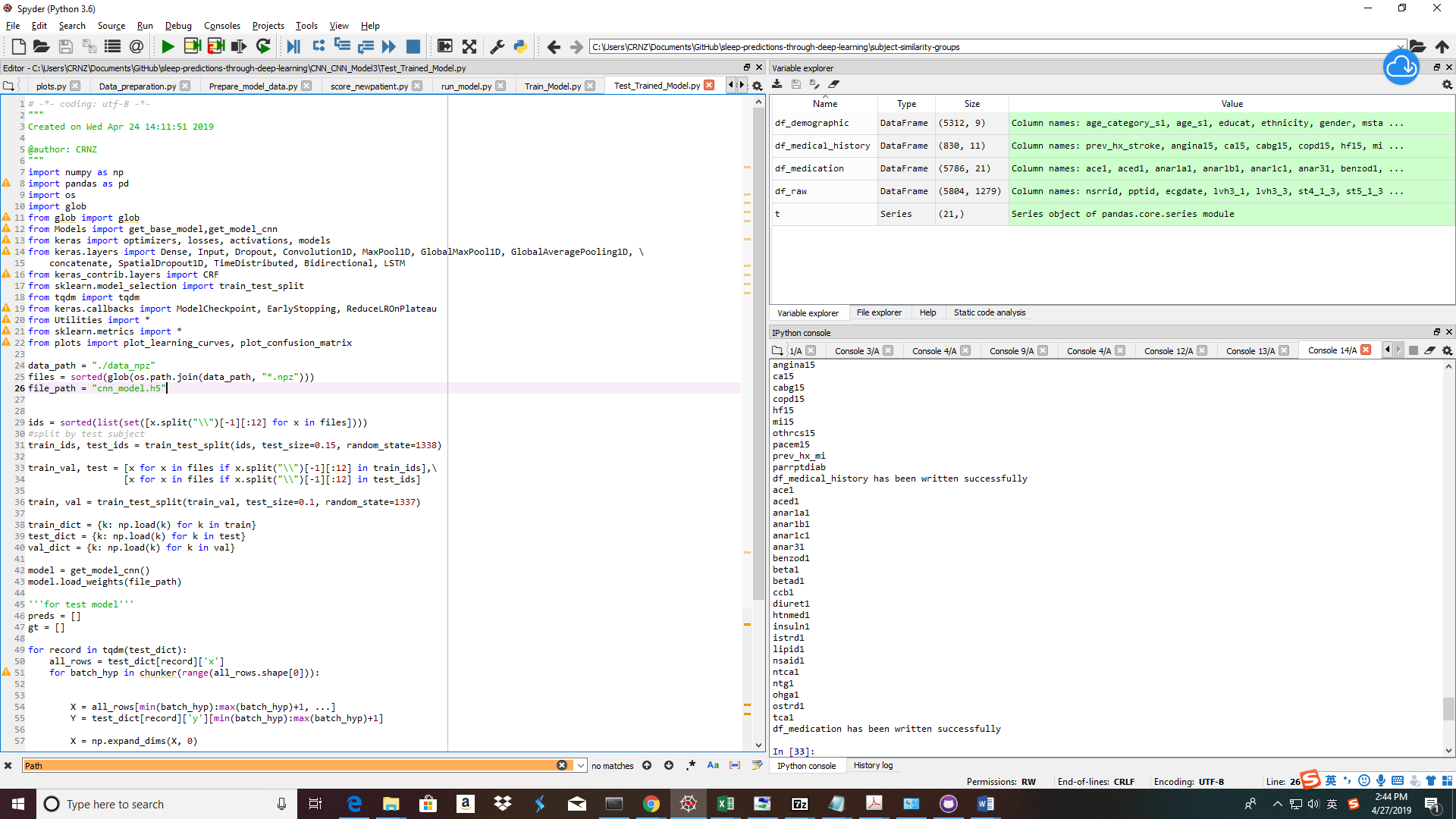


Figure 8: input required for Test\_Trained\_Model.py

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

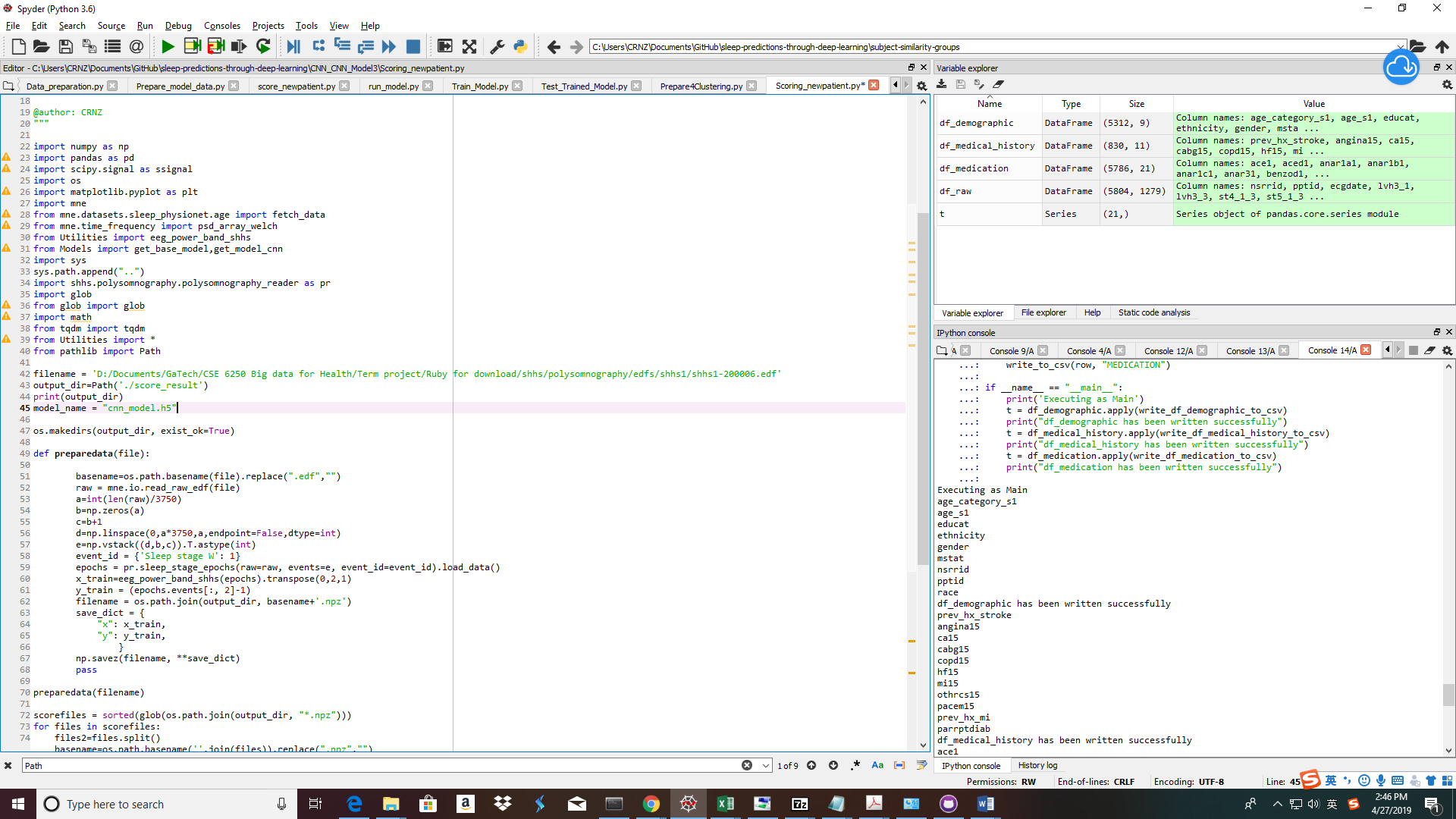
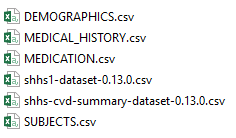


Figure 9: input required for Scoring\_newpatient.py

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

‘subject-similarity-groups/output’:

