Source Code Listing

Capstone Application

Contains the code for the GUI through which one uses the machine learning model created using the models module.

*class*capapp2.**gui**(*master: Tk*, *\*args*, *\*\*kwargs*)

User interface class for classifying DICOM files.

…

This application will control multiple pages and pass data throughout frames. This application will contain two total pages. The first page will allow the user to load the dicom files desired before making predictions. The second page will display the results of the predictions with the metadata from the gathered dicom files. The application will classify DICOM files related to mammograms or side images of breasts.

**Parameters:**

**master** (*Tk*) – The master window of the algorithm.

**download**()

Download a csv file containing basic data

**load\_images**()

Load Image Names

…

Creates a list of paths directed towards the files of interests. These paths are then used within the predict function to load the data and make predictions based on said data.

**main\_dashboard**()

Load the dashboard portion of the application.

…

Loads the dashboard with six independent plots. Each chart will depend on the data provided and will only show when they have the data required to plot.

**plot**()

Plot the main graphics of the dashboard

…

Creates six charts comprised of three pie charts and three bar graphs. These are based on the predictions, sex of the patient, age, and the imaging modality.

**predict**()

Trigger for the predict action.

…

Load the machine learning model and the related DICOM files to make predictions based on data found within the header file and image. It then saves the prediction together with the metadata on to the pandas *DataFrame* for reporting on the main dashboard page. The main dashboard page is finally loaded with all of the plots set on the top.

**start\_page**()

Page to load the user DICOM files.

…

This page will be used by the user to direct the algorithm to the desired list of DICOM files that the user would like to have analyzed. The user will then press the predict button, causing the algorithm to call the trained model to make predictions upon the DICOM files selected. Once the predictions are finished, the second page will be displayed containing basic charts.

Models Module

This file will contain all of the actualized models created from the abstract model class(es) made within the base.py file.

models.**base\_image\_classifier**(*img\_height: float*, *img\_width: float*)

Basic Image Classifier for model comparison improvement.

…

A class containing a simple classifier for any sort of image. The models stemming from this class will function to only classify the image in one manner alone (malignant or non-malignant). This model will not contain any rescaling or data augmentation to show how significant the accuracy between a model with rescaling and data augmentation is against a model without any of these.

**Parameters:**

* **img\_height** (*float*) – The height, in pixels, of the input images. This can be the maximum height of all images within the dataset to fit a varied amount that is equal or less than the declared height.
* **img\_width** (*float*) – The width, in pixels, of the input images. This can also be the maximum width of all images within the dataset to fit a varied amount that is equal or smaller in width to the declared dimension.
* **batch\_size** (*int*) – One of the factors of the total sample size. This is done to better train the model without allowing the model to memorize the data.

**Returns:**

* **inputs** (*{img\_input, cat\_input}*) – Input layers set to receive both image and categorical data. The image input contains images in the form of a 2D numpy array. The categorical input is a 1D array containing patient information. This is mainly comprised of categorical data, but some nominal data.
* **output** (*Dense Layer*) – The last layer of the model developed. As the model is fed through as the input of the next layer, the last layer is required to create the model using TensorFlow’s Model class.

models.**base\_image\_classifier2**(*img\_height: float*, *img\_width: float*)

Basic Image Classifier with rescaling and data augmentation.

…

A class containing a simple classifier for any sort of image. The models stemming from this class will include rescaling and data augmentation for the sake and purpose of normalizing the data.

**Parameters:**

* **img\_height** (*float*) – The height, in pixels, of the input images. This can be the maximum height of all images within the dataset to fit a varied amount that is equal or less than the declared height.
* **img\_width** (*float*) – The width, in pixels, of the input images. This can also be the maximum width of all images within the dataset to fit a varied amount that is equal or smaller in width to the declared dimension.
* **batch\_size** (*int*) – One of the factors of the total sample size. This is done to better train the model without allowing the model to memorize the data.

**Returns:**

* **inputs** (*{img\_input, cat\_input}*) – Input layers set to receive both image and categorical data. The image input contains images in the form of a 2D numpy array. The categorical input is a 1D array containing patient information. This is mainly comprised of categorical data, but some nominal data.
* **output** (*Dense Layer*) – The last layer of the model developed. As the model is fed through as the input of the next layer, the last layer is required to create the model using TensorFlow’s Model class.

models.**tumor\_classifier**(*img\_height: float*, *img\_width: float*, *batch\_size: int*)

Complete Tumor Classification Algorithm.

…

A class containing a simple classifier for any sort of image. The models stemming from this class will include rescaling and data augmentation for the sake and purpose of normalizing the data.

**Parameters:**

* **img\_height** (*float*) – The height, in pixels, of the input images. This can be the maximum height of all images within the dataset to fit a varied amount that is equal or less than the declared height.
* **img\_width** (*float*) – The width, in pixels, of the input images. This can also be the maximum width of all images within the dataset to fit a varied amount that is equal or smaller in width to the declared dimension.
* **batch\_size** (*int*) – One of the factors of the total sample size. This is done to better train the model without allowing the model to memorize the data.

**Returns:**

* **inputs** (*{img\_input, cat\_input}*) – Input layers set to receive both image and categorical data. The image input contains images in the form of a 2D numpy array. The categorical input is a 1D array containing patient information. This is mainly comprised of categorical data, but some nominal data.
* **output** (*Dense Layer*) – The last layer of the model developed. As the model is fed through as the input of the next layer, the last layer is required to create the model using TensorFlow’s Model class.

Pipeline Module

Algorithms used to process data before modeling.

…

A set of algorithms used to feed in and process data before used within the model. This will contain the data extraction from its rawest form and output the final form of the data set. The main source of data will be image related from the Cancer Imaging Archive.

pipeline.**extract\_data**(*filename: str*)

Extract the data from the .dcm files.

…

Reads each independent file using the pydicom library and extracts key information, such as the age, sex, ethnicity, weight of the patient, and the imaging modality used.

**Parameters:**

**filename** (*str*) – Path to the file (either relative or absolute). The file must end in .dcm or else an error will be thrown.

**Returns:**

**datapoint** – Dictionary comprised of the image data (numpy array), and the metadata associated with the DICOM file as its own separate *key:value* pair. This only pertains to the patient data and NOT the metadata describing how the image was taken.

**Return type:**

dictionary

**Raises:**

* **InvalidDicomError** – The file selected for reading is not a DICOM or does not end in .dcm. Set in place to stop the algorithm in the case that any other filetype is introduced. Causes an error to be printed and the program to exit.
* **AttributeError** – Occurs in the case that the DICOM file does not contain some of the metadata used for classifying the patient. In the case that the metadata does not exist, then the model continues on with the classification and some plots may be missing from the second page.

pipeline.**load\_data**(*filename: str*, *batch\_size: int*)

Load the data using tensorflow data set library.

…

Uses the os library and the TensorFlow Data api to load, batch, and process the data for training.

Parameter

filename*str*

Leads to a file containing the paths to all of the DICOM files as well as metadata.

batch\_size*int*

Factor of the length of the data set.

**returns:**

* **X** (*TensorFlow Dataset*) – Zipped dataset containing both image data and categorical data together.
* **y** (*TensorFlow Dataset*) – Data set containing the classifications of the data.

pipeline.**load\_data2**(*filename: str*, *batch\_size: int = 1*)

Load the DICOM data as a dictionary. …

Creates a dictionary containing three different numpy arrays. The first array is comprised of multiple DICOM images, the second contains the categorical data as a vector, and the third contains the classification in numerical form.

**Parameters:**

* **filename** (*str*) – path to a file which contains the metadata, classification, and path to the DICOM file. Will also contain some sort of ID to better identify the samples.
* **batch\_size** (*int*) – Factor of the dataset size. Currently set to one as the standard for testing purposes.

**Returns:**

**data** – Dictionary containing the encoded values for the metadata and the transformed image for input to the model.

**Return type:**

dictionary

pipeline.**transform\_data**(*datapoint: dict*)

Transform the data into an format that can be used for displaying and modeling.

…

Grabs the extracted data and begins transforming the data into a format that can be used for display in a dashboard as well as for modeling purposes.

**Parameters:**

**datapoint** (*dictionary*) – Contains the image and related metadata in *key:value* pair format.

**Returns:**

**datapoint** – same dictionary with the categorical data transformed into numerical (from text).

**Return type:**

dictionary

**Raises:**

* **AttributeError** – Indicator of the *key* does not exists.
* **KeyError** – Indicator of the *key* does not exists.