System and Unit Test Report

Product Name: Deforestation Detector

Team Name: Deforestation Detector

Team Members:

Donnovan Henry

Sam Edwards-Marsh

John Beresford

Geo Ochoa

Chris Sterza

Date: 3/8/22

System Test Scenarios

Sprint 1

Stories:

- As a user, I would like to see an indication of the loading progress of the page, because large media can take many seconds to load and I don't want to be stuck with a blank screen
- As a user, I would like the colors of the website to be fitting to the rest of the experience and aesthetically pleasing, because color can affect readability of text and visibility of other elements.
- As a user, I would like the fonts used on the page to be readable and fitting to the purpose of the website.
- As a user, I would like to be able to see more about the experience in an about page, because it will allow me to understand its purpose.

Scenario:

1. Visit http://deforestationdetector.com and you should be met with a loading screen
2. The type on the page should be legible and fully visible

3. Click on the *Learn More* button and you should be met with an about page

Scenario:

1. Go into the rainforest.ipynb and run all cells
2. User should see all possible neighbors for each image

Scenario:

1. Go into the rainforest.ipynb and run all cells
2. User can expect to see metrics on model performance for validation

Sprint 2

Stories:

- As a user, I want to be able to navigate the UI/experience intuitively, because otherwise I may miss information and/or get frustrated.
- As a user, I would like to learn more about the problem in an information page so I can understand its importance.
- As a user, I would like the data to be accurate, because misinformation can be more dangerous than ignorance.
- As a user, I would feel more comfortable browsing a page with a professional-looking icon.
- As a user, I want the data to be relevant to the purpose of the experience
- As a user, I would like to arrive at a landing page, because it would make the rest of the experience more accessible
- As a user, I would like to be able to visualize the scope of today's deforestation because it will help me to understand its threat to my future

Scenario:

1. CD into directory with rainforest.py
2. Call rainforest.py and specify a supported base model
3. Train model.
4. User should notice evaluation returns a validation accuracy of over 90%

5. User should notice that one graph appears with seventeen individual confusion matrices, one for each label
6. User should then see a sub-directory for their model under the checkpoints directory, which they can then load.

Scenario:

1. CD into directory with rainforest.py
2. Call rainforest.py and specify a supported base model
3. Initialize Docker container from the Dockerfile.
4. Train model
5. User should see a great speed increase in training time, and should see notification about graphics card usage.

Scenario:

1. After loading, the user should be met with a landing page
2. The user should click on the *explore* or the *learn more* button.
3. User should see an indicator of how to interact with the page if idle for too long on the explore view
4. Mouse interactions with the 3D environment should be intuitive (drag up/down to move the camera back/forward, drag left/right to rotate)
5. The information presented in the learn more view should be direct and not misleading. It should also be verifiable.
6. The user should be able to see an icon in the browser tab.
7. The user should be able to see and interact with the 3D visualization of the Amazon Rainforest

Sprint 3

Stories:

- As a user I want to know how much deforestation is due to human intervention rather than natural causes

- As a user, I would like to see a proper domain name, because it makes me feel like the information there is more reputable.
- As a user, I want to be able to input an image and get a prediction on it.
- As a user, I would like calls to action that can lead me to ways of contributing to the efforts against deforestation.
- As a user, I want to know which specific regions in the Amazon rainforest are being affected by deforestation.

Scenario:

1. User enters the page and clicks the *learn more* button
2. The user should be able to find information on what contributes to deforestation and, of those things, what is natural or not.
3. User then exits the learn more view and enters the explore view and clicks on an interactable element.
4. The user should then see more information about that specific element, and calls to action for contribution.

Scenario:

1. CD into directory with rainforest.py
2. Call rainforest.py and specify a supported base model
3. Initialize Docker container from the Dockerfile.
4. Train model
5. User should see a great speed increase in training time, and should see notification about graphics card usage.

Scenario:

1. CD into directory with rainforest.py
2. Call rainforest.py and specify a supported base model
3. Train model.
4. User should notice evaluation returns a validation accuracy of over 90%
5. User should notice that one graph appears with seventeen individual confusion matrices,

one for each label

6. User should then see a sub-directory for their model under the checkpoints directory, which they can then load.

Sprint 4

Stories:

- As a user, I want the elements in the experience to have pleasing micro-interactions.
- As a user, I would like clear/intuitive indication of how to navigate unique elements of the experience, because otherwise I might get lost or miss content.
- As a user, I would like to see an example of the satellite images used for the 3D experience while I am investigating it.

Scenario:

1. Navigate to repo with scai_test.py
2. Run `python scai_test.py`
3. User should see all (6) tests pass

Scenario:

1. User enters the website and is displayed an *Explore* and *Learn More* button. A mouse hover on any of these changes the button's brightness.
2. On the *Learn More* page, a list of sections should appear whose underline property changes on a mouse hover indicating these can be clicked.
3. At the bottom of both the *Learn More* and Label Pages, a list of websites are displayed whose text color property is changed to orange on a mouse hover.
4. On the Learn More, a section titled "About This Page" displays the satellite images used for the 3D experience.
5. On both the *Learn More* and Label pages, an "x" appears at the top right which displays an animation indicating we can go back to the page we were at.
6. After clicking the "x", we are back to the explore page and if we remain idle the cursor shows instructions to use the WASD or arrow keys to move around.

Unit Tests

Chris:

- **LabelState.initializeImages():** This function takes a prebuilt list of images and updates the DOM to show the images. I manually changed the image list to test the two equivalence classes:
    - An empty image list
        - The function correctly did not create any new DOM elements.
    - A nonempty image list
        - The function correctly created a DOM element for each of the images in the list.
- **LabelState.setDomElements():** This function takes a predefined list of DOM elements necessary for the state to function correctly and adds them to an array of referenceable DOM elements. I manually changed the list of DOM elements to test the three equivalence classes:
    - An empty list:
        - The function did not add or remove any elements from the referenceable DOM elements, as expected.
    - A nonempty list:
        - The function added the listed DOM elements to the referenceable DOM elements without removing any, as expected.
    - A list with a nonexistent DOM element:
        - The function added an undefined DOM element to the referenceable DOM elements without removing any, as expected.

- **LabelState.setEventHandlers():** This function sets some predefined event handlers to handle certain Label State events. I manually changed the list of event handlers to test the two equivalence classes:
    - An empty list:
        - The function didn't create any event handlers, as expected.
    - A nonempty list:
        - The function created the listed event handlers, as expected. The event handlers themselves also performed as expected.
- **LabelState.updateLabel(label):** This function updates some DOM elements with the information of the passed label. I manually changed the passed label to test the two equivalence classes:
    - An existent label:
        - The function updated all the DOM elements to match the label's info, as expected.
    - A nonexistent label:
        - The function throws an error, as expected.

John:

- **Clock.update()**
    - Browsers locked to 60fps
        - Desired behavior is inherent
    - Browsers with no fps cap
        - Ensured the update() function properly limits the internal clock to 60 updates per second at most by using fps monitoring tools and ensuring behavior is consistent at all fps >= 60
- **ViewState.setView(newView)**
    - newView is a possible view
        - Ensured that all elements in the app that trigger a transition trigger the correct transition manually
        - Ensured the newView is properly added the the view history stack
    - newView is not a possible view

        - Manually tested, error is caught and the transition is canceled silently
- **ViewState.back()**
    - There is a valid view state to backtrack to
        - Ensured the proper view is popped off the view history stack and used to trigger a transition
    - There is no valid view state to backtrack to (e.g. on landing page, have backtracked fully already)
        - Ensured the transition is canceled

Geo:

- **CursorState.setText( )**
    - Cursor text accurately with what it is passed
        - Tested manually with random strings and ensured cursor was updated with intended string
- **CursorState.update()**
    - t and view variables received the expected values
        - Manually tested performance.now()'s return value to be correct timer intervals by outputting values to console
        - Manually tested getView() to return Exploring by checking if we made it inside the conditional statement. This is further verified by the cursor being updated.
- **CursorState.handleIntersection()**
    - Cursor is accurately updated with text when hovering over a label
        - Manually tested by ensuring cursor text is updated on a hover to a label. This is made possible as the updated text is triggered by the intersection.
- **CursorState.handleViewChange()**
    - Current View is an exploring view

        - Ensured being idle on the explore page displays text over the cursor with remove('out')
    - We are not in the exploring view
        - Ensured not being in the exploring view removes the cursor with remove('in').
- **CursorState.handleMouseDown()**
    - Current View is Exploring
        - The function sets the lastMouseDown time to the current time, as expected.
    - Current view is not Exploring
        - The function returns early without setting lastMouseDown time to the current time, as expected.
- **CursorState.handleMouseUp()**
    - Current View is Exploring and Learn More is clicked
        - The function sets the new state view to "about", as expected.
    - Current View is Exploring and cursor is intersecting
        - The function sets the new state view to "investigate" as expected.
    - Current View is Exploring and cursor is not intersecting
        - The function returns without setting a new state view, as expected.
    - Current View is not Exploring
        - The function returns before setting a new state view, as expected.
- **BackdropState.setHandlers()**
    - Current View is Exploring
        - Backdrop is not removed and not visible, tested manually
    - Current View is not Exploring
        - Backdrop is added and visible, tested manually
- **AboutState.setHandlers()**
    - About page bottom shadow disappears when we scroll down
        - Manually tested by ensuring conditional only executes on a return call of LearnMore for target

        - Manually tested by outputting scrollTop's value to console and making sure the shadow disappears when the target of 12.5 is reached from scrolling.
    - About page loads in and begins at the top of the page
        - Call to function getView() returns about state, in which the about page is displayed and begins at the top
            - Manually tested by scrolling to bottom, exiting about page and revisiting the page
            - Gave variable view other states which branched to the else condition, removing the about page and ensuring we

Donovan:

- set_NLABELS():
    - Description and expectations
        - set_NLABELS is supposed to take in a dataframe. If not, the data is invalid, and the test should fail, resulting in the NLABELS global variable remaining with the value "None". Furthermore, the dataframe is meant to contain exactly two columns. First, an image_name column describing the name of the file within the directory. Second, the corresponding labels for that image. If valid input is provided to the function, then after the call to it, the NLABELS global variable will no longer be "None". This is the fact that we leveraged to test input validity. In terms of testing valid input, we supplied a substantially smaller csv file within a substantially smaller data file, where we called the set_NLABELS after setting up a dataframe object from that csv file. After this, we checked the NLABELS value to ensure that the number of labels corresponded to the number of unique labels within the csv.
    - Equivalence classes
        - Not a dataframe
        - Should and does result in "None"

            - Dataframe with less or more than 2 columns, or with a missing image_names/tags column
        - Should and does result in "None"
            - Dataframe with missing image_names column
        - Should and does result in "None"
            - Dataframe with missing tags column
        - Should and does result in "None"
            - Valid dataframe with 5 unique labels and 6 image_names
        - Shoulda and does result in 5
- f1()
    - Description and expectations
        - This is one of the functions with the caveats as mentioned above. This function as passed as an argument into the tensorflow evaluation metric. This function is bound by a relu/sigmoid activation function, which, in turn, means that it cannot be outside of the range of 0 or 1. In turn, the only meaningful tests I could think of were to ensure that the elements provided were 32-bit tensors. If so, then as mentioned, they must be bound between 0 and 1. Furthermore, we hand-tested the function. We provided some arbitrary values between 0 and 1 (for reasons described several times), and verified that their ouputs were as expected.
    - Equivalence classes
        - Non-tensor values
            - Should return None and does return None
        - Prediction close to ground truth
            - We chose values of [0.8, 0.2, 0.8] and [0.4, 0.1, 0.7] to be satisfiably close for all intents and purposes. We then hand-calculated and expected result, obtained (after rounding) 0.571, which we then obtain.
        - Prediction far from truth

            - We chose values of [0.9, 0.8, 0.4] and [0.1, 0.1, 0.6] to again be satisfiable. We again then hand-calculated this and obtained the expected result of 0.258
- reverseHot()
    - Description and expectations
        - This function is supposed to take an array of integers, each of which corresponds to a label in the specified classes array. In turn, this means two things need to be expected as arguments. First, the label_numpy array must obviously be a numpy array, but it also must be a numpy array of integers. Furthermore, if the classes list is not a list of strings, then that would also be invalid input. If either of these occur, we would return "None". However, if not, then we expect that the label string returned would be the concatenation of the elements in the classes array for each element that has a corresponding index.
    - Equivalence classes
        - Non-list classes
            - Should and do obtain a returned value of "None"
        - List classes with non-string
            - Should and do obtain a returned value of "None"
        - Non-array labels
            - Should and do obtain a returned value of "None"
        - Array labels and list classes with out of bounds index
            - Should and do obtain a returned value of "None"
        - Array labels and list classes with in bounds indices
            - Should and do obtain joining of elements in classes list with corresponding labels array element.

Sam:

- create_data():
    - Description and expectations:

        - create_data is meant to take in a dataframe object with the file names and the corresponding labels. We followed the exact same invalid input logic as the function directly above, set_NLABELS. For testing valid input, we created a dataframe off of the small csv file described above, and passed that as an argument into the create_data function. After this, the two returned values should be something other than "None". If so, we know that two data generators were successfully created. The only way that the data generators could fail to be created is if they threw an exception, or if they returned ("None", "None").
    - Equivalence classes:
        - Train datagenerator is None
        - Validation datagenerator is None
        - Train dataframe is None
            - Train and Validation datagenerators should equal None
        - Label classes are None
            - Train and Validation datagenerators should equal None
- f1loss():
    - Description and expectations:
        - This function's testing is literally the exact same as f1 just with different hand-tested, expected outcomes. This function is just a differentiable version of the f1 function such that it can be used as a loss function so the model can perform a gradient.
    - Equivalence classes:
        - Non-tensor values
            - Should return None and does return None
        - Prediction close to ground truth
            - We chose values of [0.8, 0.2, 0.8] and [0.4, 0.1, 0.7] to be satisfiably close for all intents and purposes. We then hand-calculated and expected result, obtained (after rounding) 0.4, which we then obtain.

        - Prediction far from truth
            - We chose values of [0.9, 0.8, 0.4] and [0.1, 0.1, 0.6] to again be satisfiable. We again then hand-calculated this and obtained the expected result of 0.717
- create_transfer_model():
    - Description and expectations:
        - This function is supposed to take in an architecture name specified by a string, initialize it, and set it as a base model followed by a flattening layer with a dense neural network. With hundreds of millions of neurons within each base model, testing for model equality here isn't really feasible here, so the way I saw it, there were two cases between invalid input and valid input. If the input was invalid because the element in the ARCH variable wasn't a string, or if the specified ARCH isn't in the set of architectures, then "None" is returned. Else a valid model is returned. This is further backed up by the fact that we custom defined the architecture set as a constant and thus if the user supplied a valid architecture, then without fail, it couldn't do anything other than return a valid architecture.
    - Equivalence classes:
        - Model architecture is valid, one of the tested models.
        - Model architecture is spelled wrong
        - Model architecture is "None"