Bradley University

Data Science with Autism: Autism Spectrum Disorder (ASD) Insights

Dinesh Kumar Sankar Raman
Krishna Koteswa Samudrala
Connor Steele
Amruth Raj Kumar Varanasi
Capstone Project - CS 594
Dr. Nikolopoulos
December 8, 2023

Abstract 3
Introduction 3
Problem Description 3
Journal Article Query Information Retrieval Task 4
Introduction 4
Proceeding Steps 4
Example Usage 5
Data Visualization Information Retrieval Task 7
Introduction 7
Preprocessing 7
Proceeding Steps 8
Example Usage 8
Image Machine Learning Task 10
Introduction 10
Preprocessing 10
Example Usage 12
Future Enhancements 13
Conclusion 13
Questionnaire Machine Learning Task 13
Introduction 13
Preprocessing 14
Proceeding Steps 17
Example Usage 24
Future Implications and Conclusion 26
References 29
Appendix 30
Appendix a. Code for journal article task: 30
Appendix b. Code for data visualization task: 31
Appendix c. Code for questionnaire task 33
Appendix d. Code for image task: 36

Abstract

The goal of this project is to provide an interactive educational resource for users to learn more about autism spectrum disorder (ASD). Two information retrieval tasks and two machine learning tasks will be implemented. And for future implications, all four tasks will be hosted on an interactive web application. No prior knowledge about ASD is required; all the user needs is a device that can connect to the internet and a curiosity about ASD.

Introduction

Two information retrieval tasks will be demonstrated, and two machine learning tasks will also be demonstrated. All four tasks show what is possible with data science and autism. The report provided below goes through in detail the steps taken to create and achieve these tasks with snapshots, including some snapshots of the web implementation. The website will have a menu bar with five items: Home, Find a Journal Article, Data Visualizations, Do I Have Autism: From Questionnaire, and Do I Have Autism: From Image. The home page provides a brief overview with instructions. The journal article and data visualizations tasks perform information retrieval, and the “Do I Have Autism” tasks use machine learning.

Problem Description

Despite a massive increase in the amount and rate of data being generated, despite modern technological advancements in society with artificial intelligence, machine learning, and computational power, and despite a growing number of students and learners diving into these fields, autism is still a problem humanity struggles to cure or even understand. So in the age of these advancements, this report/project aims to help bridge that gap even further, solidifying the intersection of data science and ASD.

Journal Article Query Information Retrieval Task

Introduction

This task aims to get autism-related journal article publications so users can learn more about autism. Users can filter through various criteria allowing the user to find specific articles that they are interested in. Users will find three drop down menus and after filling those, they will get the publication link.

Proceeding Steps

The dataset for this task was taken from Kaggle which is an open source platform where the pre-processed data can be found.

In this dataset the user can find the author name, year of publication, publication link, autism type and more. In this task the dataset was narrowed down to these attributes to get the link of the publication.

In this task the user is going to get three dropdown menus. The first is Author (author name) where the user can find the author who has written about the different types of autism. The second menu is Autism Type where the user will find the different types of autism and the user can select one or more at a time. And finally the third dropdown menu is Year (of publication).

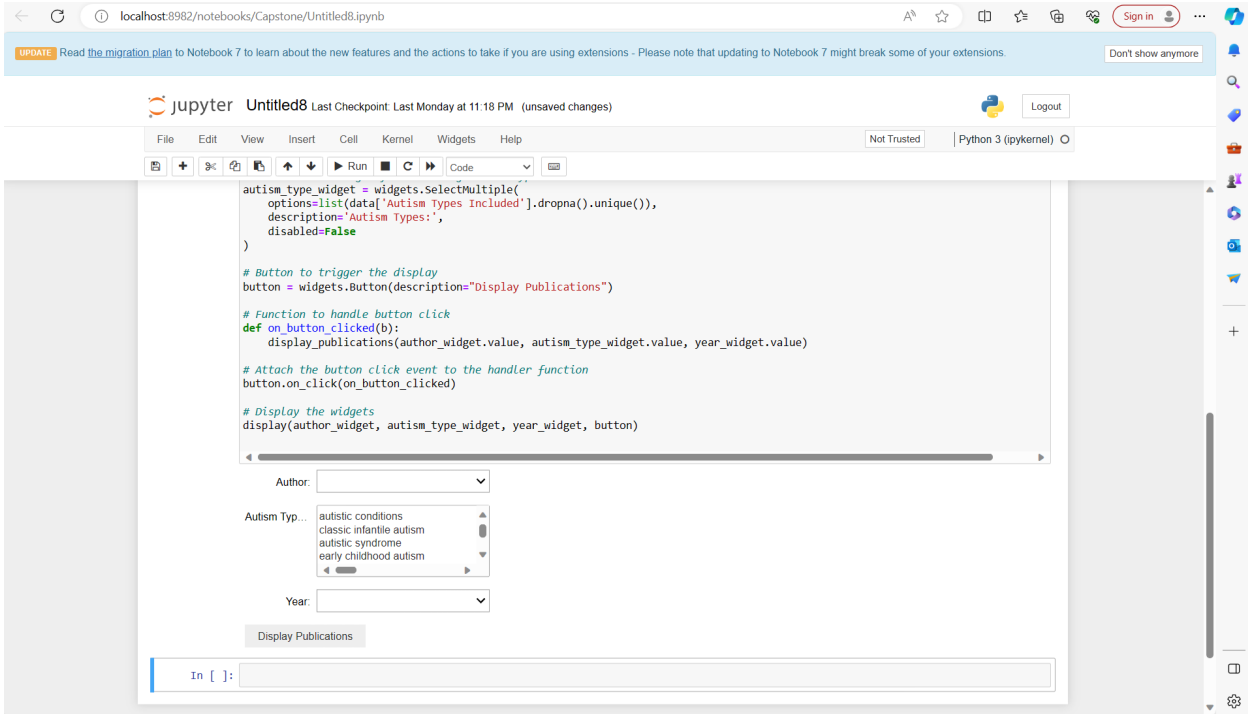
After filling all these dropdowns and hitting the Display Publication button, the link(s) for the publication(s) will output. Once clicked, the user will be redirected to that article page.

Working on the code:

- ☐ Libraries/packages: pandas, ipywidgets, Ipython.display
- ☐ Pandas are used for loading and manipulating the dataset
- ☐ Ipywidgets are used to create the dropdowns of the Author, Autism Type, and Year
- ☐ Ipython.display is used to load the publication link

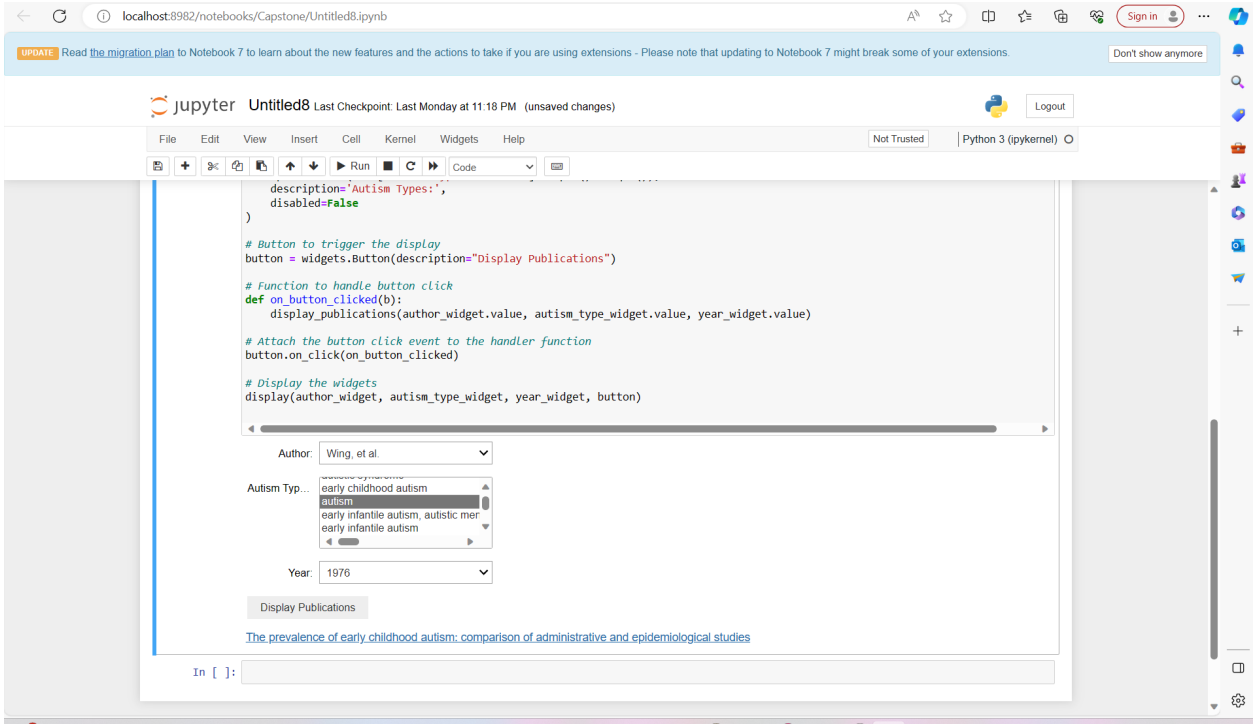
Example Usage

Running the code:



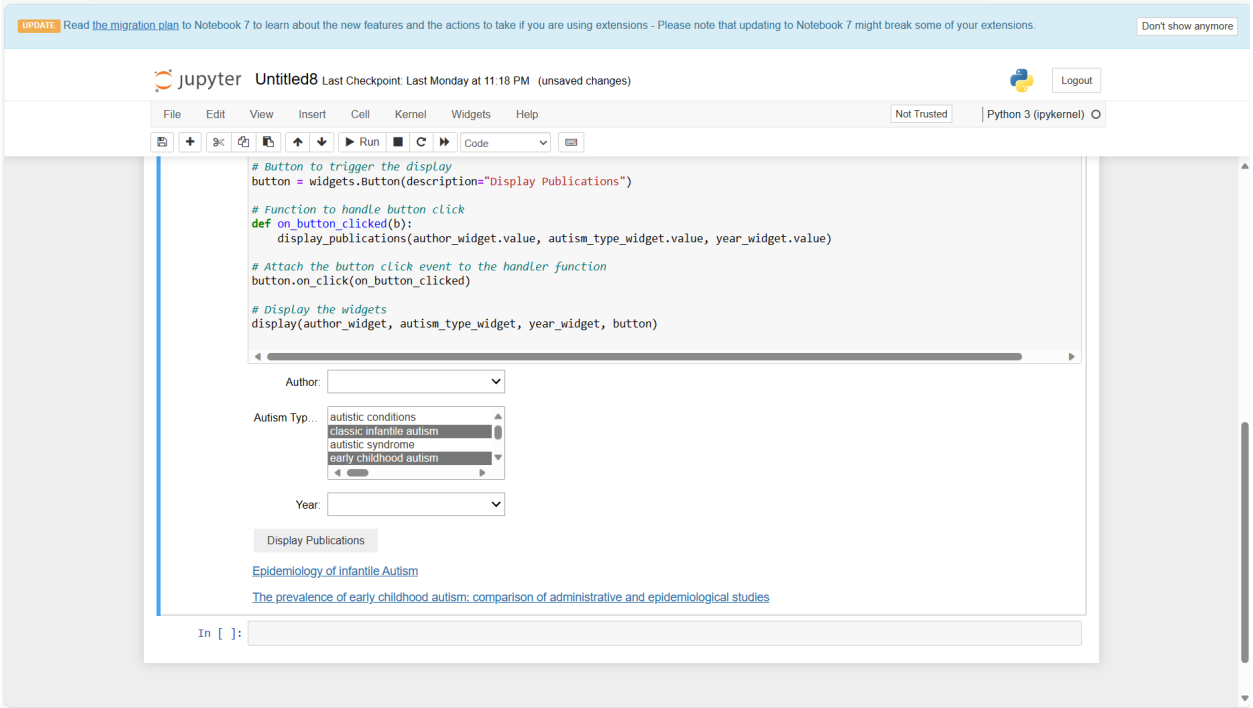
After running the user can fill all the dropdowns. Otherwise the user can select any one of the dropdowns and then hit the Display Publication button. Then the user will get the article link.

Output:



Then the user can click on the link and be redirected to the article publication page:

The user can also select one or more autism types at a time:



Data Visualization Information Retrieval Task

Introduction

The purpose of this task is to provide an interactive interface for users to learn more about fatty acid concentrations of autistic and non autistic children. The data set came from a nutritional study conducted in Arizona.

The user selects from a drop down menu what chemical compound to select. More information about the selected fatty acid also gets displayed if the user chooses to do so.

Preprocessing

In terms of how clean the datasets were, they were relatively clean. As such, all cleansing steps were easier to implement directly in Excel.

The original dataset contained a column indicating if the user was autistic or non autistic. Two separate tables were instead made, one containing all autistic and the other non-autistic. The indicator column was deleted from both and the csv files were saved and named accordingly.

The common food sources were not in any of the tables either. Kaggle did not appear to have any fatty acid/food source csv tables, nor was one readily available elsewhere. So, a new table was created in Excel with two columns, the fatty acid name and a brief description of the

sources. The sources were found from a website called American Academy of Family Physicians, but the sources are also readily available elsewhere such as webmd, National Institutes of Health, or other journal articles. A few of the common food sources were then filled in for each fatty acid, along with some quick facts.

Once these three tables were saved in the same folder, the preprocessing was complete.

Proceeding Steps

Matplotlib in python was chosen as the package and tool for the data visualization. As for the GUI implementation, tkinter was used. And later it was realized that the graphs would not clear the output when the button was pressed, causing the program to be slow once too many charts were shown. So, a new package was installed using the line “from IPython.display import display, clear_output”. This makes it so that upon the button press, the output is cleared each time.

All three csv files were loaded in, and two functions had to be written; one called show_chart for the %concentration retrieval and the other called show_sources for the food source retrieval.

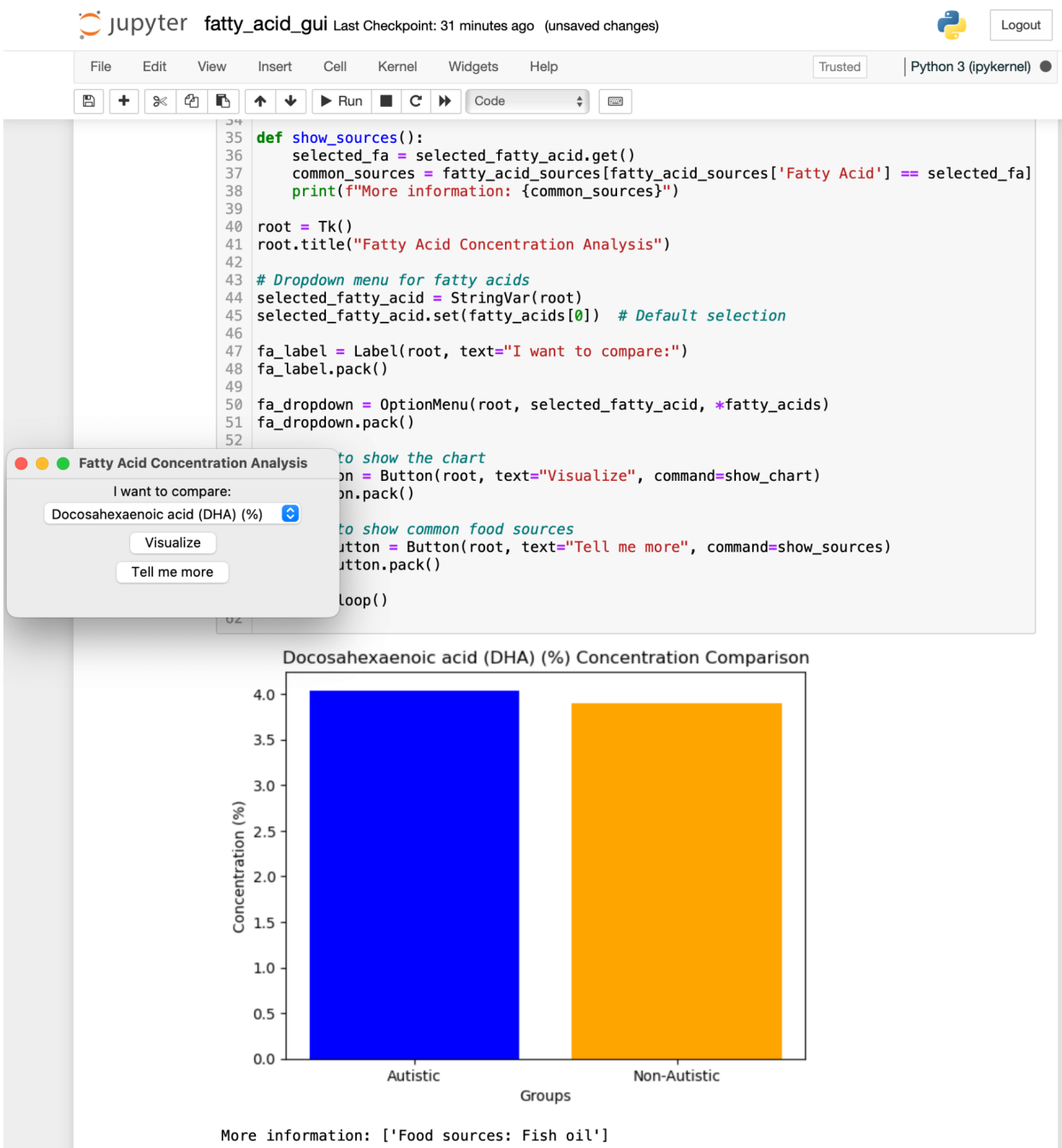
For the charts, the x and y-axis had to be relabeled depending on the fatty acid selected. For example, Oleic acid is a percentage, DHA/AA is unitless, and seafood meals per month are just “meals”. Some if-then statements fixed this issue. In addition, the average value is used, as each attribute type is numeric.

For the sources, the main issue was that the names of the fatty acids created in the food sources table had to match exactly the other tables (case sensitive).

The final steps were tweaking the visuals and behavior, and ensuring the labels were correct depending on the units and chart type.

Example Usage

Upon running the cell, the GUI opens as a new window. No output will be shown unless the user clicks “Visualize” or “Tell me more”:



From the image above, the “Visualize” button calculates the average value of the column “Docosahexaenoic acid (DHA) (%)” from both the autistic and non-autistic csv files. And after clicking “tell me more”, it shows that this fatty acid is commonly found in fish oil, pulling from the created table.

In this case the difference might be negligible between non autistic and autistic. So all in all with this task, the user gets to compare autistic and non-autistic groups and get quick facts about what each fatty acid is.

Image Machine Learning Task

Introduction

In an effort to advance early autism detection, this project develops a machine learning-based image recognition system for screening toddlers. The core objective is to build and integrate a more robust and user-friendly image recognition system that leverages the power of machine learning for autism screening, particularly focusing on a Graphical User Interface (GUI) for ease of use.

Preprocessing

Data Acquisition and Organization

The dataset for this project is sourced from Kaggle and consists of pre-processed images of toddlers. These images are categorized into three distinct sets: training, validation, and testing, with each set containing two subcategories - 'autistic' and 'non-autistic.' This classification is crucial for the supervised learning approach adopted in this project.

Data Verification and Cleanup

A critical step to ensure data integrity, involving checks for image format compatibility and corrupted files.

Data Augmentation Techniques

To enhance the model's learning and generalization capabilities, various augmentation techniques were applied to the training and validation images, including rotation, width and height shift, shear transformation, zooming, and horizontal flipping.

Data Generators

Custom data generators were developed using TensorFlow's ImageDataGenerator, tailored for each dataset category. These generators not only streamline the flow of image data into the model but also apply the necessary preprocessing and augmentation steps.

GUI Development and Integration

A significant breakthrough in this project is the development of a GUI. This interface significantly lowers the barrier to using the model by providing a simple and intuitive platform for users.

Image Browsing and Selection: Users can browse and select images of toddlers from their local storage, making the tool accessible to a wide range of users.

Real-Time Image Analysis: Upon image selection, the GUI displays the image and immediately engages the ML model for analysis.

Result Visualization: The GUI clearly displays whether the analyzed toddler image is classified as 'autistic' or 'non-autistic,' providing quick and easy-to-understand results.

Model Evaluation

Post-training, the model undergoes rigorous evaluation on the test dataset. Key metrics such as accuracy, precision, recall, and F1-score are computed. Additionally, misclassified images are examined to understand and improve the model's decision-making process.

Here is a sample output of the key evaluation metrics:

Accuracy: 0.85

Precision: 0.88

Recall: 0.82

F1-score: 0.85

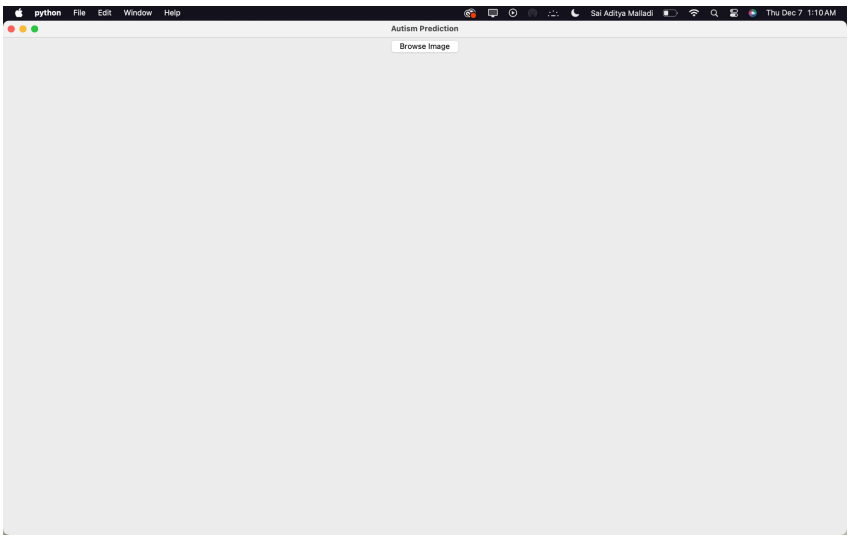
These metrics provide insights into the model's performance, where accuracy represents the overall correct classification rate, precision measures the ratio of true positives to the total predicted positives, recall quantifies the ability to correctly identify positive cases, and the F1-score balances precision and recall.

The fully trained and tested model is saved in HDF5 format, facilitating easy integration with the GUI and ensuring readiness for practical deployment. The saved model can be seamlessly loaded into the GUI for real-time analysis.

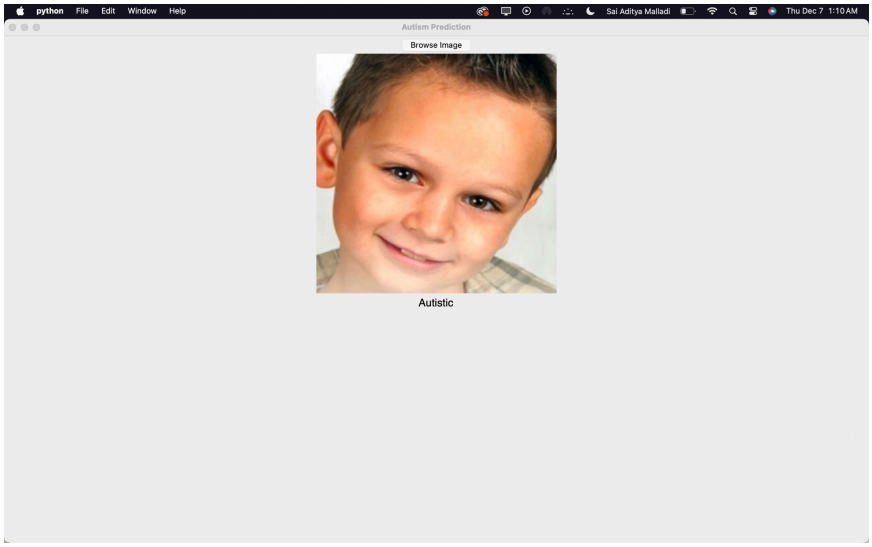
Example Usage

This project represents an initial step towards integrating sophisticated machine learning models with intuitive graphical user interfaces (GUIs) for medical applications, specifically focusing on

autism detection in toddlers. While the current version demonstrates promising potential, it's important to note that the system is in its early stages, primarily due to the limited size of the dataset, which comprises approximately 2,500 images.



The above image is the GUI.



The above image shows the GUI working, the ML model tells the child in the image as autistic.

Future Enhancements

Future enhancements are crucial to elevate the tool’s effectiveness and reliability. Key areas of focus for improvement include:

Dataset Expansion: The current dataset's limited size is a significant constraint on the model's ability to learn and generalize. Expanding the dataset with more images would significantly improve the model's training, leading to better accuracy and a more robust screening tool.

Model Refinement with Advanced Techniques: Implementing advanced techniques like transfer learning could enhance the model's performance. Transfer learning involves using a pre-trained model on a large dataset and fine-tuning it for the specific task of autism detection, which can be particularly beneficial given the current dataset's size constraints.

Professional Feedback and Iterative Development: Collaborating with medical professionals to gather feedback and incorporate their insights can greatly improve the model's diagnostic accuracy and relevance. This feedback loop is essential for aligning the tool with clinical needs and standards.

Web Platform Integration: To increase accessibility and user reach, deploying the tool on a web platform is a promising direction. This would make the tool more accessible to a broader audience, including healthcare professionals and caregivers, facilitating early screening and intervention.

Acknowledging Current Limitations: It's crucial to acknowledge the system's current limitations in accuracy due to the early stage of development and limited data. Users should be advised to use the tool as a preliminary screening method, not a definitive diagnostic solution.

Conclusion

In conclusion, while the current system represents a meaningful stride in leveraging AI for pediatric healthcare, it's just the beginning. Continuous development, informed by a larger dataset, advanced machine learning techniques, and professional medical input, is essential to realize its full potential as a reliable resource in autism detection.

Questionnaire Machine Learning Task

Introduction

The dataset shows a sample of results from adults (aged 18 and up) who answered some questions in a mobile app called ASDTests. The app screens autism in people of all ages (using its own machine learning algorithm and different sets of questions depending on the age group),

but the app itself has not been reviewed well. Many claim it isn't accurate. Furthermore, the app is no longer available on the iOS store.

Other studies have been done using these questionnaires as well, such as here: https://pubmed.ncbi.nlm.nih.gov/36643888/. One task this study does is rank which questions are most predictive of autism.

Since the underlying algorithm of the app isn't publicly available to test and edit, this study gave the team an idea to instead create a rule-based question and answer system using a decision tree. This will be faster than going through all 10 questions, and will provide the additional insight as to which questions may be more important and indicative of ASD than others. The test is for adults aged 18 and up.

The dataset is from Kaggle and did not directly provide the source link to the questionnaire. But in the dataset description in Kaggle it says: "In this dataset, we record ten behavioural features (AQ-10-Adult)". Another Kaggle dataset sourced a researcher named Dr. Thabtah, and used the autism research centre's ("arc") "Q-Chat-10" questionnaire for a toddler dataset, citing "arc" and the University of Cambridge. Dr. Thabtah is also sourced in this adult dataset and is formatted the same as the toddler one, including "arc" and the same university, so it's assumed this questionnaire is used: (https://docs.autismresearchcentre.com/tests/AQ10.pdf).

Each question is listed below:

1. I often notice small sounds when others do not
2. I usually concentrate more on the whole picture, rather than the small details
3. I find it easy to do more than one thing at once
4. If there is an interruption, I can switch back to what I was doing very quickly
5. I find it easy to 'read between the lines' when someone is talking to me
6. I know how to tell if someone listening to me is getting bored
7. When I'm reading a story I find it difficult to work out the characters' intentions
8. I like to collect information about categories of things (e.g. types of car, types of bird, types of train, types of plant etc)
9. I find it easy to work out what someone is thinking or feeling just by looking at their face
10. I find it difficult to work out people's intentions

Preprocessing

The dataset was first loaded into WEKA to check for abnormalities: Each attribute was selected one by one to quickly check for missing values, outliers or other problems. It was then shown that two columns contained issues, the "ethnicity" column and the "age" column. The "ethnicity" column contained holes, and the "age" column contained holes and outliers.

For the "ethnicity" column, it was determined that python would be best suited for cleaning. This column corresponds to the "contry_of_res" column, so each hole was filled on a case-by-base

basis based on the “contry_of_res” column using this website as a source: https://www.nationmaster.com/country-info/stats/People/Ethnic-groups. Each hole was filled with the most common ethnicity per country.
Example code of this process is shown below:

```
import pandas as pd

Autism = pd.read_csv('Autism_screening_data.csv')

# Replace 'ethnicity' values for rows where 'contry_of_res' is __
Autism.loc[Autism['contry_of_res'] == 'Egypt', 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == "'United Arab Emirates'", 'ethnicity'] = 'Middle E
Autism.loc[Autism['contry_of_res'] == 'Lebanon', 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == 'Afghanistan', 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == "'Saudi Arabia'", 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == 'Jordan', 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == 'Russia', 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == "'New Zealand'", 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == 'Brazil', 'ethnicity'] = 'Latino'
Autism.loc[Autism['contry_of_res'] == "'United Kingdom'", 'ethnicity'] = 'White-European
Autism.loc[Autism['contry_of_res'] == 'Iran', 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == "'United States'", 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == 'Pakistan', 'ethnicity'] = 'Asian'
Autism.loc[Autism['contry_of_res'] == 'Iraq', 'ethnicity'] = 'Middle Eastern'
Autism.loc[Autism['contry_of_res'] == "'United Arab Emirates'", 'ethnicity'] = 'Black'
Autism.loc[Autism['contry_of_res'] == 'Austria', 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == 'Argentina', 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == 'Bahamas', 'ethnicity'] = 'Black'
Autism.loc[Autism['contry_of_res'] == "'Hong Kong'", 'ethnicity'] = 'Asian'
Autism.loc[Autism['contry_of_res'] == 'Japan', 'ethnicity'] = 'Asian'
Autism.loc[Autism['contry_of_res'] == 'Ukraine', 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == 'Kazakhstan', 'ethnicity'] = 'White-European'
Autism.loc[Autism['contry_of_res'] == 'Azerbaijan', 'ethnicity'] = 'Middle Eastern'
```

This was then updated into the original csv file and checked to verify that the holes have been filled:

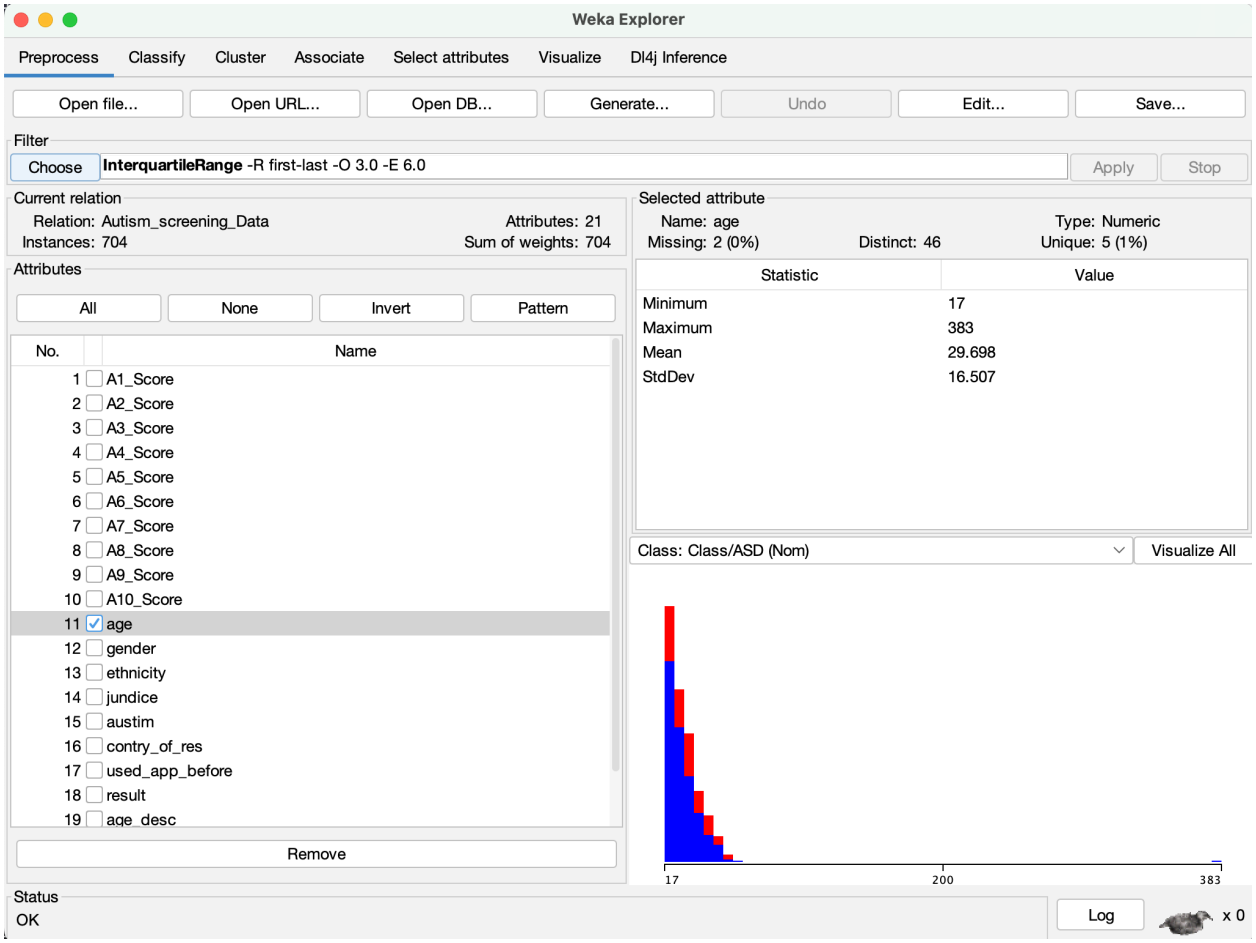
```
# Write the updated DataFrame back to the CSV file
Autism.to_csv('Autism_screening_data.csv', index=False)
```

```
ethnicity_missingvalues_count = Autism['ethnicity'].isnull().sum()
ethnicity_missingvalues_count
```

```
0
```

This was loaded into WEKA for further verification that the holes were gone.

Then WEKA showed that the “age” column had a max value of 383 and 2 missing values:



This value was located and deleted in the csv file before replacing all null values with the average value back in python. Finally this was loaded back into the csv file:

```python
average_age = Autism['age'].mean()

# Display the average age
print("Average age:", average_age)


# Replace null values in the 'age' column with the average age
Autism['age'].fillna(average_age, inplace=True)
```

```
Average age: 29.19400855920114
```

```python
# Write the updated DataFrame back to the CSV file
Autism.to_csv('Autism_screening_data.csv', index=False)
Autism

```

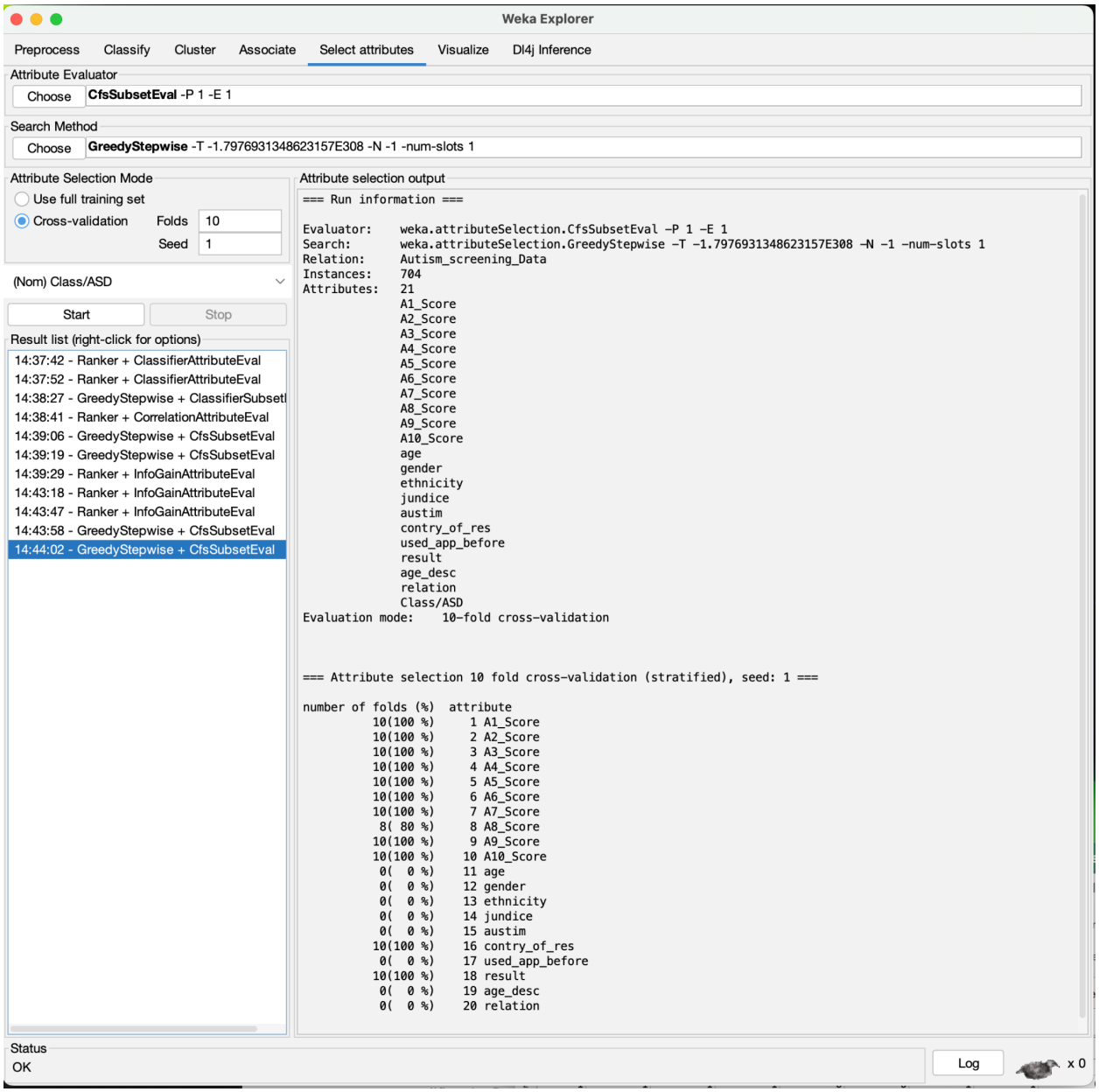
It's shown that deleting the outlier caused the average age to go down by about 1 year (when rounding).

As far as answers to the questionnaire, the values have already been converted into boolean (1 or 0) values, which will make the decision tree's rules easier to implement. In the key of the questionnaire it says: "Score 1 point for Definitely or Slightly Agree on each of items 1, 7, 8, and 10. Score 1 point for Definitely or Slightly Disagree on each of items 2, 3, 4, 5, 6, and 9". It's assumed to put 0 otherwise.

Finally, the csv file was loaded back into WEKA and checked for any remaining holes or extreme values. The dataset was then determined to be ready for attribute selection.

Proceeding Steps

Attribute selection in WEKA was used to see if the cleansed dataset could be reduced for simplification when predicting the class value, "Class/ASD". Multiple attribute evaluators and search methods were used. Most ranked the questionnaire questions the highest, along with country of residence and ethnicity. In addition, most of the 10 questions themselves were equally likely to correlate with the class attribute:



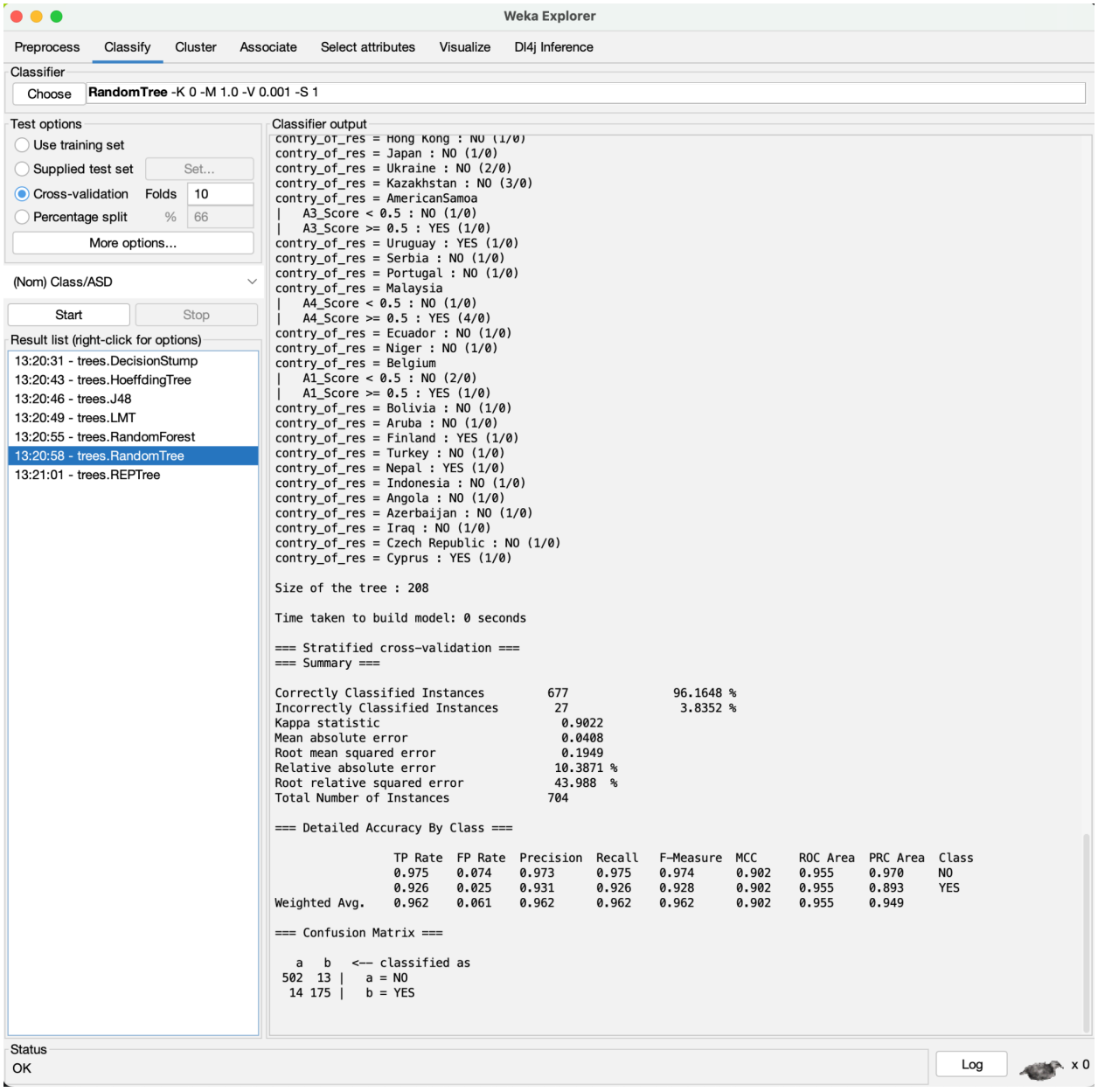
CfsSubsetEval and GreedyStepwise with 10 fold cross-validation were ultimately used to select the attributes. As seen above, all attributes with 0% number of folds were dropped; leaving the questions themselves (A1-A10), country of residence, and result (the sum of the questions). However the team also wanted to know which questions were more important than others, so “results” was deleted. If “results” was used, most decision trees’ algorithms were similar to this, which is close to what the original questionnaire said. For reference it says if the individual scores 6 or above, consider referring them for a specialist diagnostic assessment. And according to these two algorithms in WEKA:

DecisionStump and REPTree: if result <=6.5: no, else yes.

J48: if result <=6: no, else yes.

But again, this doesn't indicate which questions or other attributes may be more insightful than others.

Shown below is the output for RandomTree:



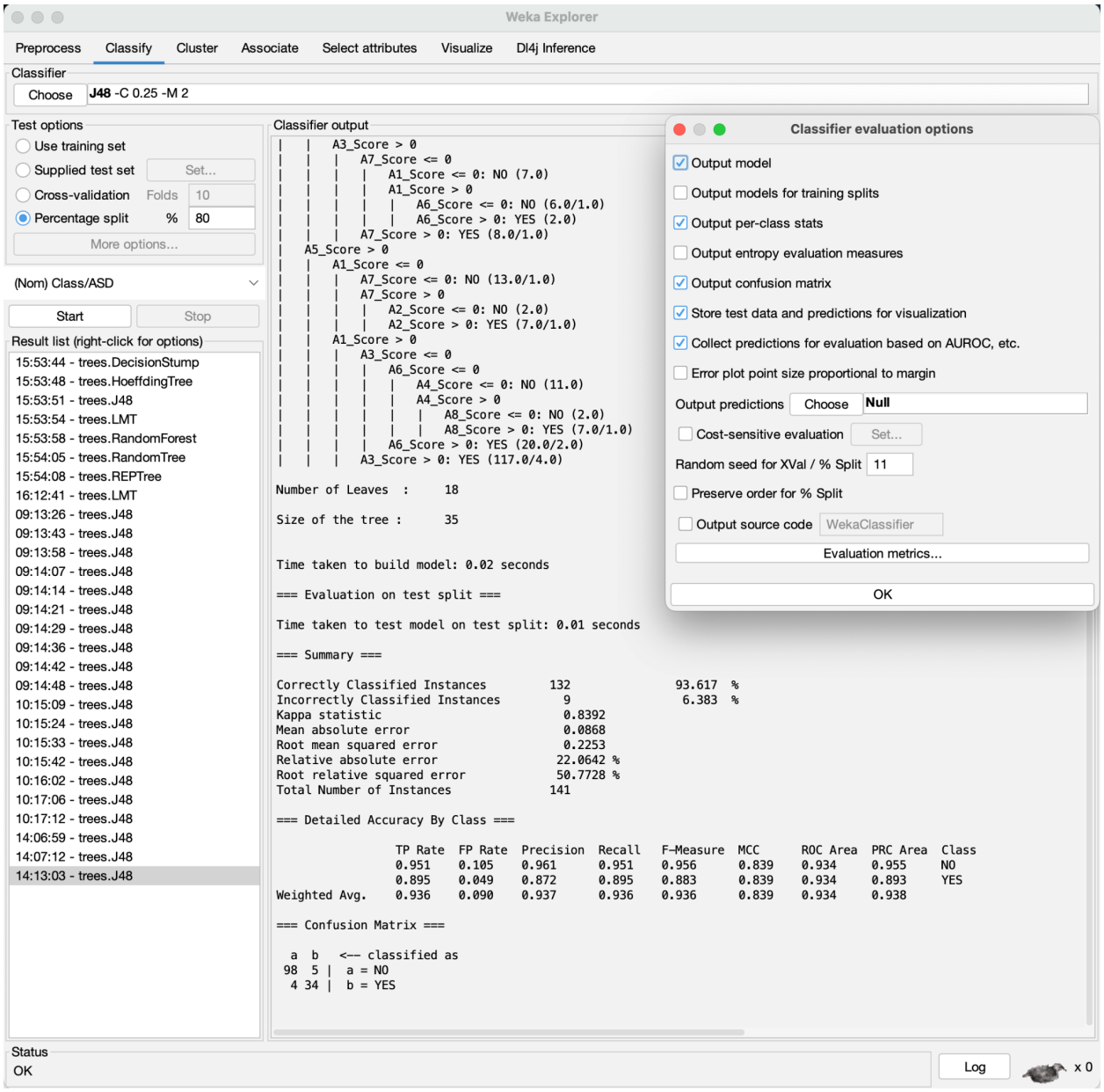
So when only counting for questions 1-10 and country of residence, all built-in decision tree models in WEKA were implemented. A summary table below shows the root mean squared error of all models using 10 fold cross-validation:

Table 1. Comparing Root Mean Squared Error With Various Models (10 fold cross validation)

Classifier	RMSE
trees.DecisionStump	0.3431
trees.HoeffdingTree	0.1713
trees.J48	0.2653
trees.LMT	0.103
trees.RandomForest	0.2376
trees.RandomTree	0.3249
trees.REPTree	0.3409

After consideration, it was ultimately decided to use the J48 algorithm. The other algorithms either had too complex of a rule-based tree (even if they had lower RMSE), didn't use the A1-A10 questions at all, or incorrectly classified too many instances when compared to others. These factors led to the decision of using J48.

To see if this model could be improved, 10 instances were run using 80/20 train test splits with the RMSE depicted below, each corresponding to a random seed number. Shown below is random seed 11 with corresponding output at 0.2253 RMSE:

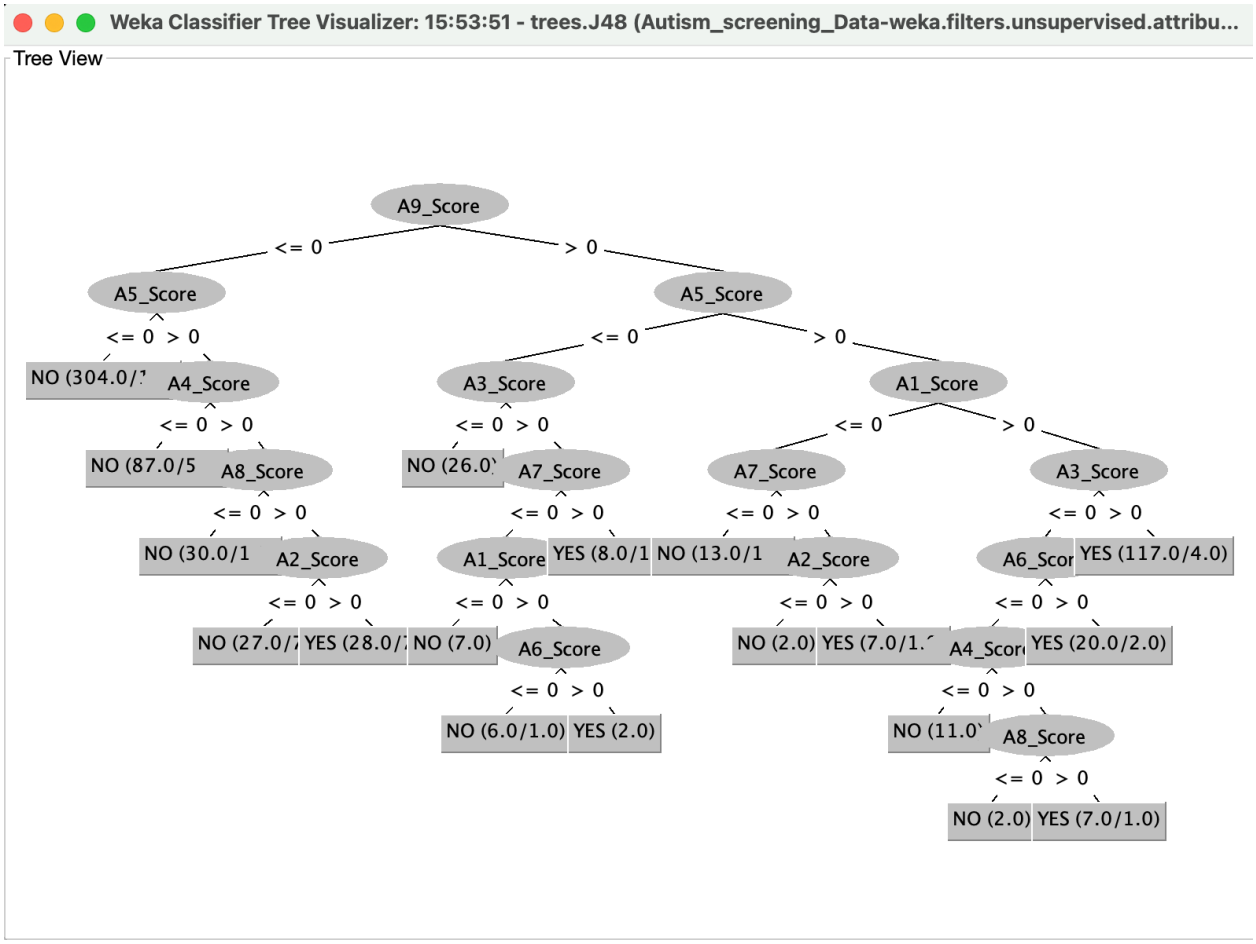


A summary of seeds 1-11 is shown below with their RMSE.

Table 2. trees.J48 80% Train 20% Test Splits

Trial (Random Seed No.)	RMSE
1	0.317
2	0.3212
3	0.3145
4	0.2941
5	0.2675
6	0.2596
7	0.2646
8	0.2837
9	0.2817
10	0.247
11	0.2253
12	0.2839
13	0.2107
14	0.231

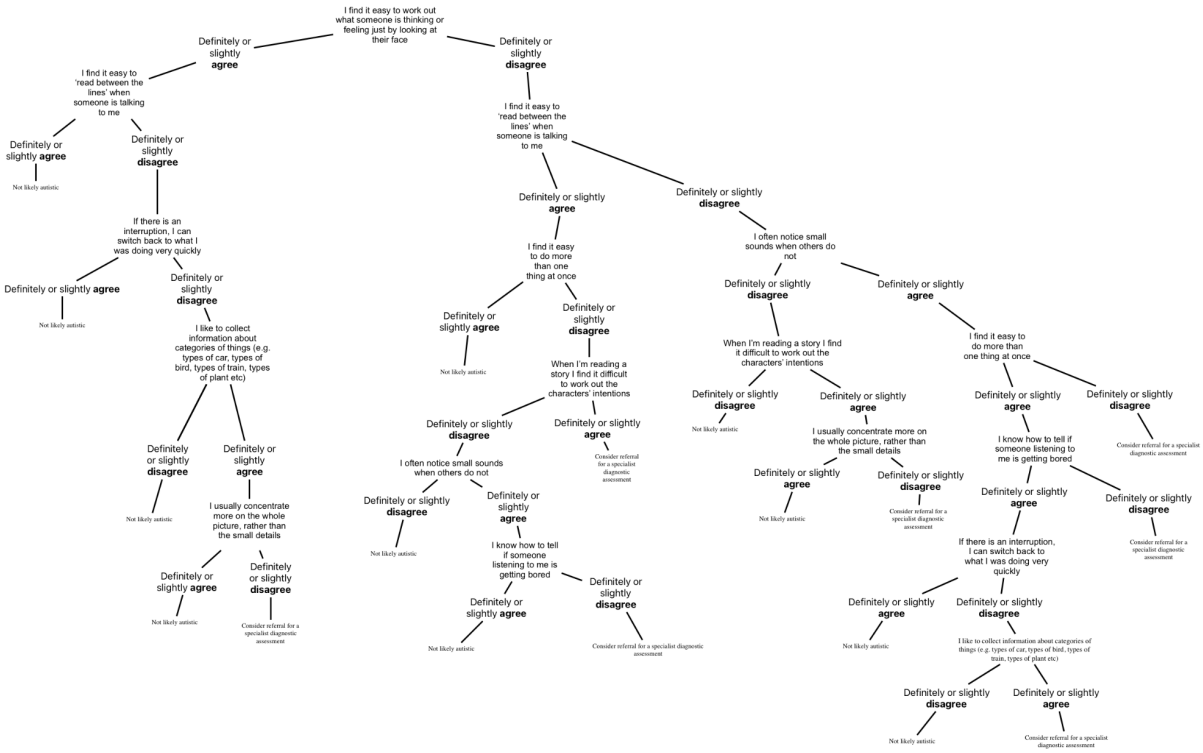
As shown above, it appears that the RMSE converges a couple times at 0.2253 and 0.2107, corresponding to random seed #11 and #13. Regardless, all trials showed the same set of rules for the tree:



Some post processing steps were then taken to make the tree easier to look at and follow. For example, consider the original questionnaire key's "rules":

"SCORING: Only 1 point can be scored for each question. *Score 1 point for Definitely or Slightly Agree on each of items 1, 7, 8, and 10. Score 1 point for Definitely or Slightly Disagree on each of items 2, 3, 4, 5, 6, and 9.* If the individual scores **6 or above**, consider referring them for a specialist diagnostic assessment."

To make this easier to follow for the user, the above WEKA output of the tree was drawn, replacing the 0s and 1s to match the key and filling in the actual questions:



Final J48 Decision Tree

Now, in terms of the web application, users can follow along themselves without having to reference the key.

The decision tree indicates the highest certainty questions at the top, and the least certain questions going down. This achieves the goal of understanding which questions are more important than others, providing additional insights for autism research.

Example Usage

According to the model, starting at the top node, if the user finds it easy to work out what someone is thinking or feeling just by looking at their face, and they find it easy to “read between the lines” when someone is talking to them, they are not likely autistic and do not need further questioning.

The decision tree was then implemented into prolog as a rule based question and answer system: An example is shown below:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- consult("~/Desktop/Data Science Resources/Capstone/Other files/questionnaire.pl").
Warning: /Users/connorsteele/Desktop/Data Science Resources/Capstone/Other files/questionnaire.pl:8:
Warning:    Singleton variable in branch: A5
Warning:    Singleton variable in branch: A4
Warning:    Singleton variable in branch: A8
Warning:    Singleton variable in branch: A2
Warning:    Singleton variable in branch: A3
Warning:    Singleton variable in branch: A7
Warning:    Singleton variable in branch: A1
Warning:    Singleton variable in branch: A6
For questions 1,7,8,10: Type "1" for definitely or slightly agree, or "0" for definitely or slightly disagree.
For questions 2,3,4,5,6,9: Type "0" for agree and "1" for disagree.

A9: I find it easy to work out what someone is thinking or feeling just by looking at their face: 0.
A5: I find it easy to 'read between the lines' when someone is talking to me: |: 1.
A4: If there is an interruption, I can switch back to what I was doing very quickly: |: 1.
A8: I like to collect information about categories of things: |: 1.
A2: I usually concentrate more on the whole picture, rather than the small details: |: 1.
Consider referral for a specialist diagnostic assessment.
Press '1' to start over, or any other key to exit: |: 1.

true.

?- start.
For questions 1,7,8,10: Type "1" for definitely or slightly agree, or "0" for definitely or slightly disagree.
For questions 2,3,4,5,6,9: Type "0" for agree and "1" for disagree.

A9: I find it easy to work out what someone is thinking or feeling just by looking at their face: 0.
A5: I find it easy to 'read between the lines' when someone is talking to me: |: 1.
A4: If there is an interruption, I can switch back to what I was doing very quickly: |: 0.
Not likely autistic.
Press '1' to start over, or any other key to exit: |:
```

Python was also used, with 1 being agree and 0 being disagree for all questions instead. Sample output:

```
                print("Not likely autistic.")
            else:
                print("Consider referral for a specialist diagnostic assessment.")
        else:
            print("Consider referral for a specialist diagnostic assessment.")

# Run the function
autism_diagnosis()
```

```
Please answer the following questions with '1' for agree or '0' for disagree.
I find it easy to work out what someone is thinking or feeling just by looking at their face (1/0): 1
I find it easy to 'read between the lines' when someone is talking to me (1/0): 0
If there is an interruption, I can switch back to what I was doing very quickly (1/0): 0
I like to collect information about categories of things (e.g. types of car, types of bird, types of train, types o
f plant etc) (1/0): 1
I usually concentrate more on the whole picture, rather than the small details (1/0): 1
Not likely autistic.
```

Future Implications and Conclusion

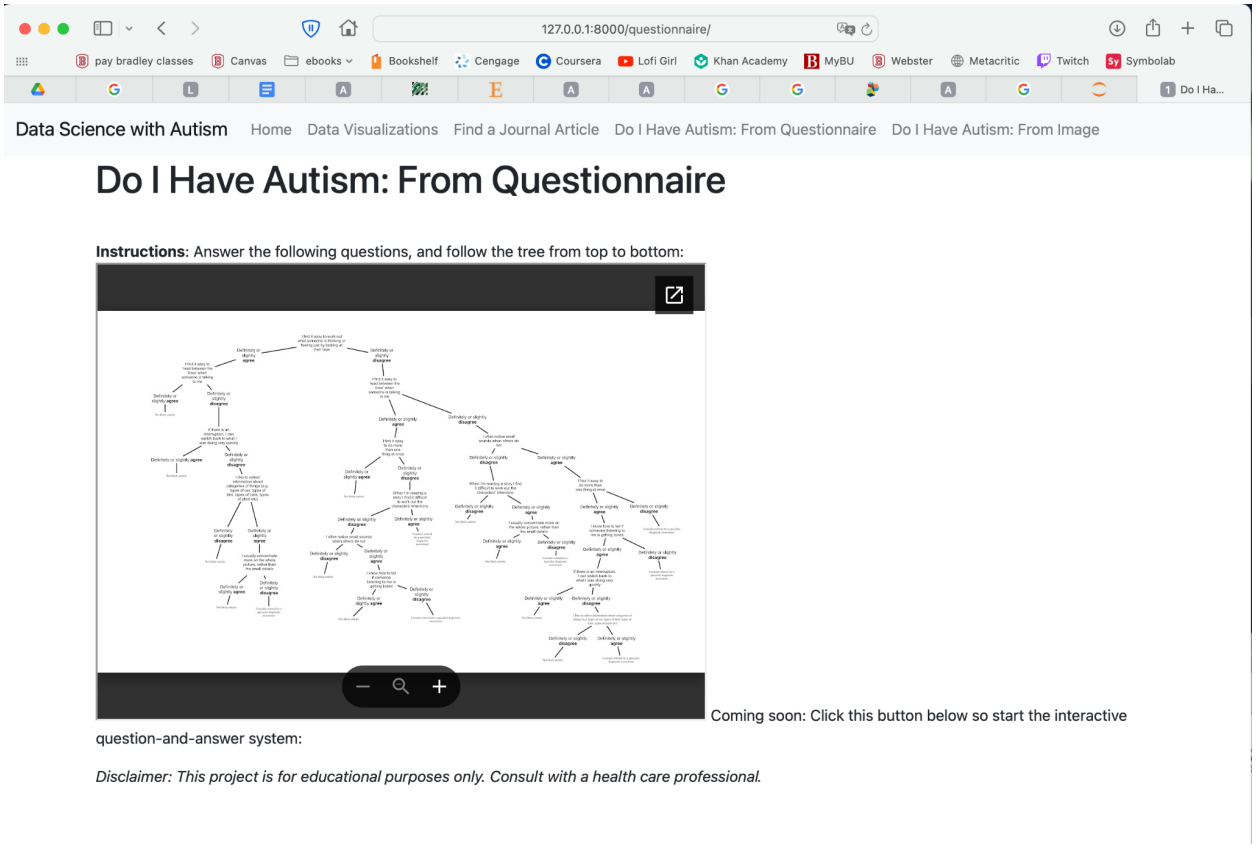
Improvements can be made on the accuracy of the various machine learning algorithms (reducing errors). And for the information retrieval tasks, more data could be used, giving users more options. For example in the journal article query, more journal articles would allow a wider variety of results. This dataset also needed to be cleansed because there were many holes, limiting the dataset even further.

The questionnaire task could also be improved by using some of the more complex decision trees, as they correctly classified more instances. But this makes coding the rule based system more difficult as there are many more branches and sub-branches.

Web scraping could also be implemented to get more data, as “complete” data sets regarding autism research appeared to be scarce outside of a few resources and kaggle.

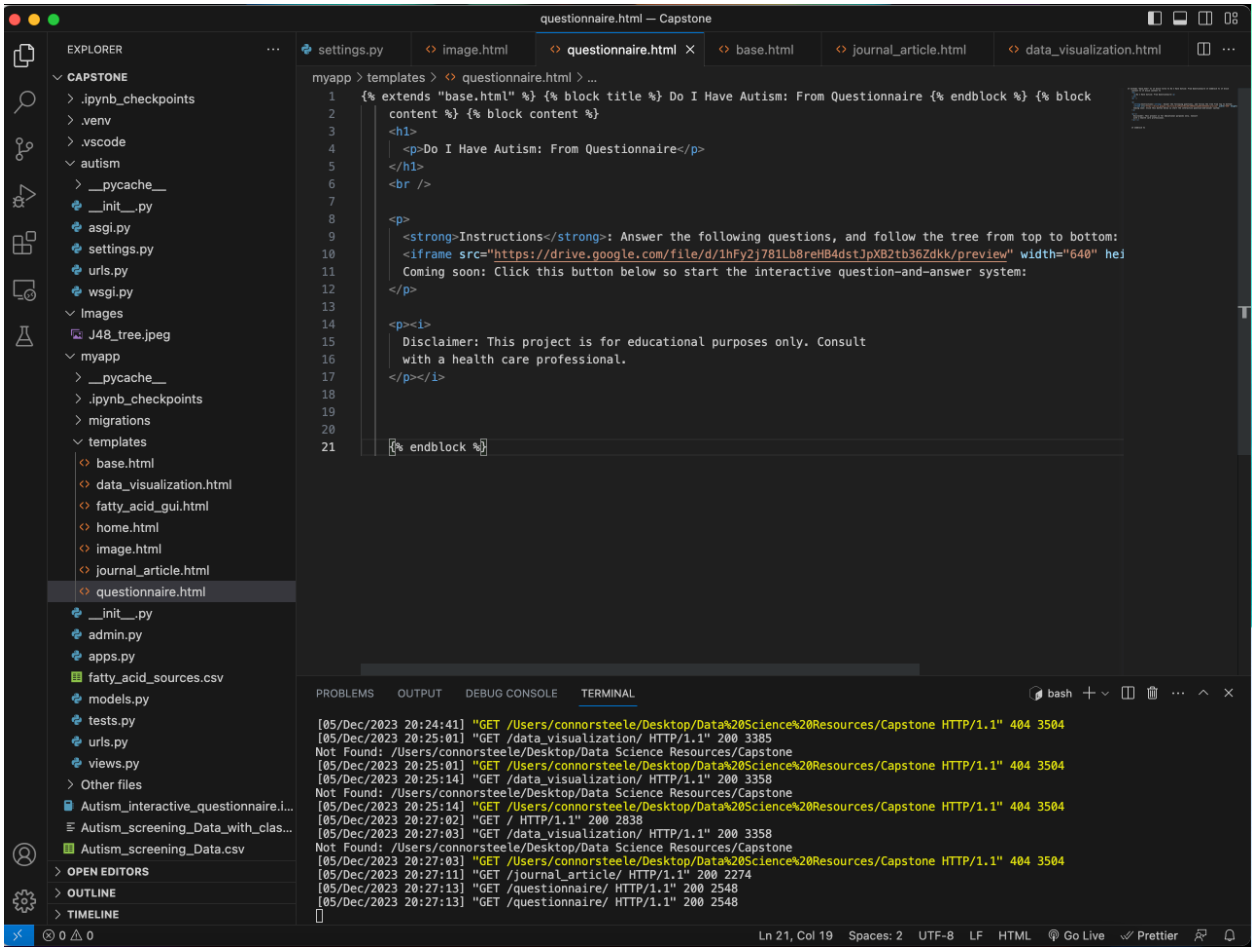
In addition to improvements to the four tasks, hosting a fully interactive website with the four tasks will reach a wider audience and will be able to unify these four tasks. This will allow for a “one-stop” resource for autism with data science.

As for the web implementation, the tool used was django in python using Visual Studio Code. Shown below is a snapshot of the “Do I Have Autism: From Questionnaire” page running locally:



This is the only “interactive” task so far. The user can scroll around and zoom into the tree and follow it visually. The other pages only have textual information.

Shown below is the HTML code for that page in Visual Studio Code:



So, the foundation is laid for the interactive website.

All in all, these four tasks aim to bridge the gap between autism prevalence and data science. An interactive web page like this will make these tasks even more accessible and easy for users, as they won't see any code and will only see GUI's and front-end visuals.

References

Centers for Disease Control and Prevention. (2022, April 7). *Autism prevalence studies data table*. Centers for Disease Control and Prevention. https://www.cdc.gov/ncbddd/autism/data/autism-data-table.html

Clinical data sets related to ASD research. Clinical Data Sets related to ASD research | Hahn Research Group. (n.d.). https://www.hahnresearchgroup.com/downloads/clinical-data-sets-related-asd-research

Dihomo-γ-linolenic acid. Dihomo-γ-linolenic acid - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/dihomo-gamma-linolenic-acid

Faizunnabi. (2018, February 12). *Autism screening*. Kaggle. https://www.kaggle.com/datasets/faizunnabi/autism-screening/data

GeeksforGeeks. (2020, November 26). *Dropdown menus - tkinter*. GeeksforGeeks. https://www.geeksforgeeks.org/dropdown-menus-tkinter/

Khan, I. (2022, March 15). *Autistic children facial dataset*. Kaggle. https://www.kaggle.com/datasets/imrankhan77/autistic-children-facial-data-set

National Institute for Health Research. (2012). PDF. https://docs.autismresearchcentre.com/tests/AQ10.pdf

NationMaster. (n.d.). *Countries compared by people > ethnic groups. international statistics*. NationMaster.com. https://www.nationmaster.com/country-info/stats/People/Ethnic-groups

Tech With Tim. (2023, Jul 29). *Learn Django in 20 Minutes!!* [Video]. YouTube. https://www.youtube.com/watch?v=nGIg40xs9e4

White, B. (2009, August 15). *Dietary fatty acids*. American Family Physician. https://www.aafp.org/pubs/afp/issues/2009/0815/p345.html

Appendix

Appendix a. Code for journal article task:

Python Code:

```
import pandas as pd
import ipywidgets as widgets
from IPython.display import display, HTML

# Load the dataset
file_path = 'for_journal_article_query.xlsx'
data = pd.read_excel(file_path)

def display_publications(selected_author, selected_autism_types, selected_year):
    filtered_data = data

    # Apply filters based on selection
    if selected_author:
        filtered_data = filtered_data[filtered_data['Author'].str.contains(selected_author, na=False,
case=False)]
    if selected_autism_types:
        autism_type_filter = filtered_data['Autism Types
Included'].str.contains('|'.join(selected_autism_types), na=False, case=False)
        filtered_data = filtered_data[autism_type_filter]
    if selected_year:
        filtered_data = filtered_data[filtered_data['Year Published'] == selected_year]

    if filtered_data.empty:
        display(HTML("<p>No matching publications found.</p>"))
    else:
        for _, row in filtered_data.iterrows():
            display(HTML(f"<a href='{row['Link to Publication']}' target='_blank'>{row['Title']}</a>"))

# Dropdown widgets for selecting Author and Year
author_widget = widgets.Dropdown(options=['] + list(data['Author'].dropna().unique()),
description='Author:')
year_widget = widgets.Dropdown(options=['] + list(data['Year Published'].dropna().unique()),
description='Year:')

# Multi-select widget for selecting Autism Types
autism_type_widget = widgets.SelectMultiple(
    options=list(data['Autism Types Included'].dropna().unique()),
    description='Autism Types:',
```

```
    disabled=False
)

# Button to trigger the display
button = widgets.Button(description="Display Publications")

# Function to handle button click
def on_button_clicked(b):
    display_publications(author_widget.value, autism_type_widget.value, year_widget.value)

# Attach the button click event to the handler function
button.on_click(on_button_clicked)

# Display the widgets
display(author_widget, autism_type_widget, year_widget, button)
```

Appendix b. Code for data visualization task:

Python code:

```
import pandas as pd
import matplotlib.pyplot as plt
from tkinter import *
from IPython.display import display, clear_output

# Load the CSV files
autistic_data = pd.read_csv('fatty_acid_autistic.csv')
non_autistic_data = pd.read_csv('fatty_acid_non_autistic.csv')
fatty_acids = autistic_data.columns.tolist()
fatty_acid_sources = pd.read_csv('fatty_acid_sources.csv')

def show_chart():
    selected_fa = selected_fatty_acid.get()

    # Bar chart
    categories = ['Autistic', 'Non-Autistic']
    values = [autistic_data[selected_fa].mean(), non_autistic_data[selected_fa].mean()]

    plt.bar(categories, values, color=['blue', 'orange'])

    plt.xlabel('Groups')

    # Update y-axis label based on the selected fatty acid
    if selected_fa.lower() == 'seafood meals per month':
        plt.ylabel('Number of Meals')
```

```
    elif selected_fa.lower() in ['dha/aa', 'epa/aa', 'n-3/n-6', 'pufa']:
        plt.ylabel('')
    else:
        plt.ylabel('Concentration (%)')

    plt.title(f'{selected_fa} Comparison')

    # Print common food sources
    common_sources = fatty_acid_sources[fatty_acid_sources['Fatty Acid'].str.lower() ==
selected_fa.lower()]['Common Sources'].values
    print(f"Common Food Sources: {common_sources}")

    # Show the chart
    clear_output(wait=True)
    plt.show()

def show_sources():
    selected_fa = selected_fatty_acid.get()
    common_sources = fatty_acid_sources[fatty_acid_sources['Fatty Acid'] ==
selected_fa]['Common Sources'].values
    print(f"More information: {common_sources}")

root = Tk()
root.title("Fatty Acid Concentration Analysis")

# Dropdown menu for fatty acids
selected_fatty_acid = StringVar(root)
selected_fatty_acid.set(fatty_acids[0])  # Default selection

fa_label = Label(root, text="I want to compare:")
fa_label.pack()

fa_dropdown = OptionMenu(root, selected_fatty_acid, *fatty_acids)
fa_dropdown.pack()

# Button to show the chart
show_button = Button(root, text="Visualize", command=show_chart)
show_button.pack()

# Button to show common food sources
sources_button = Button(root, text="Tell me more", command=show_sources)
sources_button.pack()

root.mainloop()
```

Appendix c. Code for questionnaire task

Prolog code:

```
%type start to start
start :-
   write('For questions 1,7,8,10: Type "1" for definitely or slightly agree, or "0" for definitely or slightly disagree.'), nl,
   write('For questions 2,3,4,5,6,9: Type "0" for agree and "1" for disagree.'), nl,
   nl,
   autism_diagnosis.
% Decision tree rules
autism_diagnosis :-
   ask('A9: I find it easy to work out what someone is thinking or feeling just by looking at their face', A9),
   (
      % A9 is 0
      (A9 = 0, ask('A5: I find it easy to \'read between the lines\' when someone is talking to me', A5), A5 = 0) -> write('Not likely autistic');
      (A9 = 0, A5 = 1, ask('A4: If there is an interruption, I can switch back to what I was doing very quickly', A4), A4 = 0) -> write('Not likely autistic.');
      (A9 = 0, A5 = 1, A4 = 1, ask('A8: I like to collect information about categories of things', A8), A8 = 0) -> write('Not likely autistic.');
      (A9 = 0, A5 = 1, A4 = 1, A8 = 1, ask('A2: I usually concentrate more on the whole picture, rather than the small details', A2), A2 = 0) -> write('Not likely autistic.');
      (A9 = 0, A5 = 1, A4 = 1, A8 = 1, A2 = 1) -> write('Consider referral for a specialist diagnostic assessment.');

      % A9 is 1
      (A9 = 1, A5 = 0, ask('A3: I find it easy to do more than one thing at once', A3), A3 = 0) -> write('Not likely autistic.');
      (A9 = 1, A5 = 0, A3 = 1, ask('A7: When I'm reading a story I find it difficult to work out the characters' intentions', A7), A7 = 0, ask('A1: I often notice small sounds when others do not', A1), A1 = 0) -> write('Not likely autistic');
      (A9 = 1, A5 = 0, A3 = 1, A7 = 0, A1 = 1, ask('A6: I know how to tell if someone listening to me is getting bored', A6), A6 = 0) -> write('Not likely autistic.');
      (A9 = 1, A5 = 0, A3 = 1, A7 = 0, A1 = 1, A6 = 0) -> write('Consider referral for a specialist diagnostic assessment.');
      (A9 = 1, A5 = 0, A3 = 1, A7 = 1) -> write('Consider referral for a specialist diagnostic assessment.');

      (A9 = 1, A5 = 1, ask('A1: I often notice small sounds when others do not', A1), A1 = 0, A7 = 0) -> write('Not likely autistic.');
```

```
        (A9 = 1, A5 = 1, A1 = 0, A7 = 1, ask('A2: I usually concentrate more on the whole picture,
rather than the small details', A2), A2 = 0) -> write('Not likely autistic.');
        (A9 = 1, A5 = 1, A1 = 0, A7 = 1, A2 = 1) -> write('Consider referral for a specialist
diagnostic assessment.');

        (A9 = 1, A5 = 1, A1 = 1, A3 = 0, A6 = 0, ask('A4: If there is an interruption, I can switch
back to what I was doing very quickly', A4), A4 = 0) -> write('Not likely autistic.');
        (A9 = 1, A5 = 1, A1 = 1, A3 = 0, A6 = 0, A4 = 1, ask('A8: I like to collect information about
categories of things', A8), A8 = 0) -> write('Not likely autistic.');
        (A9 = 1, A5 = 1, A1 = 1, A3 = 0, A6 = 0, A4 = 1, A8 = 1) -> write('Consider referral for a
specialist diagnostic assessment.');

        (A9 = 1, A5 = 1, A1 = 1, A3 = 0, A6 = 1) -> write('Consider referral for a specialist
diagnostic assessment.');
        (A9 = 1, A5 = 1, A1 = 1, A3 = 1) -> write('Consider referral for a specialist diagnostic
assessment.');
        true
    ),
    restart.

autism_diagnosis :- restart.

% Predicate to ask a question and read the user response
ask(Question, Answer) :-
    write(Question), write(': '), read(Answer).

% Ask if the user wants to start over
restart :-
    nl,
    write('Press \'1\' to start over, or any other key to exit: '),
    read(Choice),
    (
        Choice = '1' -> start;
        true
    ).


:- start.
```

Python code:

```
def autism_diagnosis():
    print("Please answer the following questions with '1' for agree or '0' for disagree.")

    # Question A9
```

```
    a9 = int(input("I find it easy to work out what someone is thinking or feeling just by looking at
their face (1/0): "))

    if a9 == 1:
        # Question A5
        a5 = int(input("I find it easy to ‘read between the lines’ when someone is talking to me (1/0):
"))

        if a5 == 1:
            print("Not likely autistic.")
        else:
            # Question A4
            a4 = int(input("If there is an interruption, I can switch back to what I was doing very
quickly (1/0): "))

            if a4 == 1:
                print("Not likely autistic.")
            else:
                # Question A8
                a8 = int(input("I like to collect information about categories of things (e.g. types of car,
types of bird, types of train, types of plant etc) (1/0): "))

                if a8 == 0:
                    print("Not likely autistic.")
                else:
                    # Question A2
                    a2 = int(input("I usually concentrate more on the whole picture, rather than the
small details (1/0): "))

                    if a2 == 1:
                        print("Not likely autistic.")
                    else:
                        print("Consider referral for a specialist diagnostic assessment.")
    else:
        # A9 disagree, A5 agree, A3 disagree, A7 agree
        a5 = int(input("I find it easy to ‘read between the lines’ when someone is talking to me (1/0):
"))
        if a5 == 1:
            a3 = int(input("I find it easy to do more than one thing at once (1/0): "))
            if a3 == 0:
                a7 = int(input("When I’m reading a story I find it difficult to work out the characters’
intentions (1/0): "))
                if a7 == 1:
                    print("Consider referral for a specialist diagnostic assessment.")
```

```
            else:
                print("Not likely autistic.")
        else:
            print("Not likely autistic.")
    else:
        # A9 disagree, A5 agree, A3 disagree, A7 disagree, A1 agree, A6 disagree
        a1 = int(input("I often notice small sounds when others do not (1/0): "))
        if a1 == 0:
            a6 = int(input("I know how to tell if someone listening to me is getting bored (1/0): "))
            if a6 == 1:
                print("Not likely autistic.")
            else:
                print("Consider referral for a specialist diagnostic assessment.")
        else:
            print("Consider referral for a specialist diagnostic assessment.")

# Run the function
autism_diagnosis()
```

Appendix d. Code for image task:

```
!pip install tensorflow opencv-python matplotlib scikit-learn
from google.colab import drive
drive.mount('/content/drive')

import os
import cv2
import imghdr
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix

base_dir = '/content/drive/My Drive/Archive'
image_exts = ['jpeg', 'jpg', 'bmp', 'png']
categories = ['train', 'valid', 'test']
subcategories = ['autistic', 'non-autistic']

# Data augmentation for training and validation
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
rescale=1./255,
rotation_range=40,
width_shift_range=0.2,
```

```
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest')

val_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

# Check and preprocess images
for category in categories:
for subcategory in subcategories:
dir_path = os.path.join(base_dir, category, subcategory)
if os.path.exists(dir_path):
for image in os.listdir(dir_path):
image_path = os.path.join(dir_path, image)
try:
img = cv2.imread(image_path)
tip = imghdr.what(image_path)
if tip not in image_exts:
print(f'Image not in ext list: {image_path}')
# os.remove(image_path) # Uncomment to enable deletion
except Exception as e:
print(f'Issue with image: {image_path}')
# os.remove(image_path) # Uncomment to enable deletion

# Create data generators
train_dir = f'{base_dir}/train'
val_dir = f'{base_dir}/valid'
test_dir = f'{base_dir}/test'

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(256, 256),
batch_size=32,
class_mode='binary')

val_generator = val_datagen.flow_from_directory(
val_dir,
target_size=(256, 256),
batch_size=32,
class_mode='binary')

# Build and compile the model
model = tf.keras.models.Sequential([
```

```
tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(256, 256, 3)),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Conv2D(16, (3, 3), activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(256, activation='relu'),
tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir='/content/logs')

# Train the model
history = model.fit(
train_generator,
epochs=20,
validation_data=val_generator,
callbacks=[tensorboard_callback])

# Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()

# Test the model
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
test_dir,
target_size=(256, 256),
batch_size=32,
```

```
class_mode='binary',
shuffle=False)
predictions = model.predict(test_generator)
predicted_classes = (predictions > 0.5).astype(int).flatten()
true_classes = test_generator.classes

# Evaluate and print metrics
print("Class Indices from Generator:", test_generator.class_indices)
print("Confusion Matrix:")
print(confusion_matrix(true_classes, predicted_classes))
print("Classification Report:")
print(classification_report(true_classes, predicted_classes,
target_names=list(test_generator.class_indices.keys())))

# Visualize misclassified images
misclassified_indices = np.where(predicted_classes != true_classes)[0]
print("Sample Incorrect Predictions:")
for idx in misclassified_indices[:5]: # Display first 5 misclassified images
img_path = test_generator.filepaths[idx]
img = tf.keras.preprocessing.image.load_img(img_path, target_size=(256, 256))
plt.imshow(img)
plt.title(f"True: {true_classes[idx]}, Predicted: {predicted_classes[idx]}")
plt.show()

# Save the model
model_path = '/content/models/imageclassifier.h5' # Specify the path with .h5 extension
model.save(model_path)
```