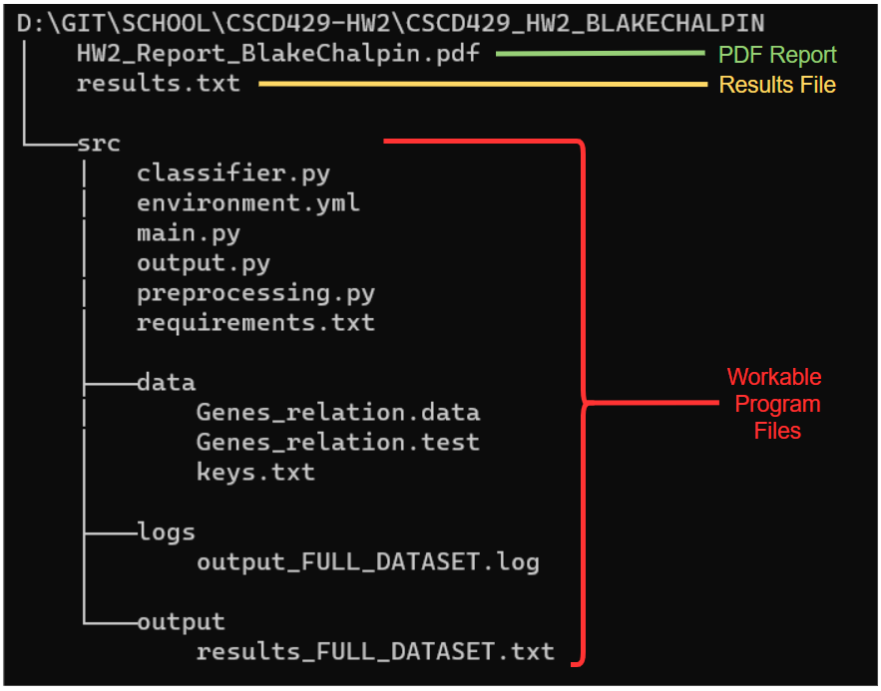
CSCD 429-040 HW 2

Blake Chalpin

5 November 2021

Project Structure

Below is the file structure for the zip file deliverable containing the code, report, and results:



Running The Program

NOTE: Before running any of the following commands below, make sure that the current working directory is `./src`

Using Conda

Using Conda, run the following commands to create the environment to run the program in (this will handle the Python version and packages):

```
$conda env create -f environment.yml
$conda activate cscd429-hw2-blake-chalpin
```

Once the Conda environment is created and activated, run the program with the command:

```
$python main.py
```

Without Conda

If you do not have Conda installed, the Python packages may still be installed using the following command:

```
$pip install -r requirements.txt
```

Note: Python 3.8 is used for this project

Once the Python packages are installed, run the program with the command:

```
$python main.py
```

Methods Used

Handling Missing Data

This program does not handle missing data. The missing data is only converted from the "?" value to NaN (Python's Null value).

Classification Method

The following is the pseudo-code for the KNN classification method that is implemented:

```
knn(train data, test data, number of neigbors = 3) {
    for each test tuple in test data {

        for each train tuple in train data {

            calculate distance between test tuple and train tuple using
                custom distance method

            store the distance and the "Localization" value for the train tuple

        }

        get 3 tuples with the smallest distance from the stored
            (distances, Localization) // this is for 3 nearest neighbors

        get the majority vote from the 3 closest tuples

        store the majority vote as the prediction for the test tuple
    }

    return predictions for test tuples
}
```

The following is the pseudo-code for the custom distance measure calculation method that is used to determine the "nearest neighbors:"

```
calculate_distance(test_tuple, train_tuple) {

    initialize distance = 0

    for each column in the tuples {

        if test_tuple[column] is NULL, or train_tuple[column] is NULL {

            add 0.5 to the distance

        }

        else if test_tuple[column] does not match train_tuple[column] {

            add 1 to distance

        }
    }

    return distance
}
```

Accuracy Calculation

The following is the pseudo-code for the method used to calculate the accuracy of the model from the output of the `knn` method:

```
calculate_accuracy(test data with labels, predictions) {

    initialize sum of correct predictions = 0

    for each prediction {

        if prediction matches label in test data {

            add 1 to sum of correct predictions

        }
    }

    accuracy = sum of correct predictions / total number of test tuples

    return accuracy
}
```

Performance

Accuracy

The overall accuracy of the 3 Nearest Neighbors (KNN with $k = 3$) classification model using the entire training data set to predict our test set is **45.779%**.

Results File

The results of our prediction are stored in the "results file" name `results.txt`

Project Issues

Program Running Time

When predicting our entire test set ($\approx$ 2000 tuples) using the entire training set ($\approx$ 8500 tuples), this program has a measured running time of $\approx$ 3.25 hours. This long running time is very unexpected and is most likely due to the data structure that I have loaded the training and test set into. To get around this we can undersample the training data set, but this will result in a different model accuracy score.

This problem can be solved by translating the code into R for a lesser program running time, but I did not have the time to do so.