Mirabel Chin Written Report

A. Project Overview

- I am trying to answer the question: is a potential exoplanet confirmed or a false positive? And can I predict which one it will be?

- I used a dataset from Kaggle called Kepler Exoplanet Dataset. I mainly focused on the koi_disposition column which has a value of either 0, 1 or 2. 0 means it is a false positive exoplanet, 1 means it is a potential exoplanet, and 2 which means it is a confirmed exoplanet.

B. Data Processing

- I loaded the CSV into rust by first creating a struct to deserialize each CSV and defined the datatypes to make it easier to work with and to keep my code organized. I just used the csv crate and Reader as well as serde Deserialize.

- I had to do a couple of transformations to make the data work. After I read the CSV in, I separated the different features (all other data except for koi_disposition) into one vector, and the labels for each column into another vector. I then iterated through the read CSV to convert all the values of 2 into 1, so that I would only be comparing 1 and 0 instead of 2 and 0. I also added some extra helpers to make sure that there are no missing values and that the array was the right shape. My second function was to extract all the 1's from the dataset and put them into a separate dataset to pass through the decision tree.

C. Code Structure

1. **Modules and Functions**

    - I have 4 modules, one for reading the csv in and doing basic data cleaning (reader.rs), one for training and making the decision tree model (trainer.rs), one for conducting all the tests (tests.rs), and the main module. I covered the reader.rs module in the data processing section.
      The trainer.rs module first calls the linfa and NDarray crates. I also created a struct called ModelReport to keep all of my code organized. It also made it easier to access the accuracy and f1 score which I return later. First, I have my training function which takes the features and labels as inputs and returns the ModelReport struct I defined earlier. The main purpose is to take the features and labels, make a dataset, train a decision tree based on the data, and return the ModelReport with the Decision tree, and other fields I outlined in the struct. Also, I first had to convert the labels from u8 to usize for linfa's Dataset. The second function, predict, takes the DecisionTree model created above and the feature data of a dataset and returns a vector of predictions.
      Lastly, for the main.rs module. I read in the CSV and called the 2 → 1 conversion function, stored the labels into a vector, and trained the model on the original dataset. I then returned the statistics of the model, opting for accuracy and F1 score. I chose accuracy since it's a standard measure, but I also included F1 score as it is a more specific way to measure the performance of the model because it measures both precision and recall (positive instances and ability to avoid positive instances). Then I called the prediction function to run the dataset of exoplanet candidates through the model. I chose to output a .txt file instead,

because I wanted a report of all the planets, but it flooded the terminal so I couldn't see the Decision Tree model statistics.

D. Tests

```
    Finished `test` profile [unoptimized + debuginfo] target(s) in 1.73s
     Running unittests src/main.rs (target/debug/deps/project_code-6c9d28000f0e6927)

running 2 tests
test tests::tests::test_data_loading ... ok
test tests::tests::test_model_pipeline ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 3.35s
```

- I had two tests in the tests.rs module. The first test checks the basic features of the dataset for the reader function (number of rows and number of columns and that features and labels have the same number of rows). It also checks the 2 → 1 conversion function as well as the 1 extraction function. The second test checks the DecisionTree training to make sure the model statistics are valid and checks that the predictions are either 0 or 1.

E. Results

- All program outputs (screenshots or pasted).

```
[/opt/app-root/src/DS210Project/project_code]
$ cargo run main
   Compiling project_code v0.1.0 (/opt/app-root/src/DS210Project/project_code)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.72s
     Running `target/debug/project_code main`

Model Evaluation:
Accuracy: 98.09%
F1 Score: 0.2927

Predictions written to 'exoplanet_predictions.txt'.
```

```
1    Predictions for koi_disposition == 1
2    Format: PlanetIndex → Predicted Class
3    Planet 0 → FalsePositive
4    Planet 1 → Confirmed
5    Planet 2 → FalsePositive
6    Planet 3 → Confirmed
7    Planet 4 → Confirmed
8    Planet 5 → Confirmed
9    Planet 6 → Confirmed
10   Planet 7 → FalsePositive
11   Planet 8 → FalsePositive
12   Planet 9 → Confirmed
13   Planet 10 → Confirmed
```

```
1147   Planet 1144 → FalsePositive
1148   Planet 1145 → FalsePositive
1149   Planet 1146 → FalsePositive
1150   Planet 1147 → FalsePositive
1151   Planet 1148 → Confirmed
1152   Planet 1149 → FalsePositive
1153   Planet 1150 → Confirmed
1154   Planet 1151 → FalsePositive
1155   Planet 1152 → Confirmed
1156   Planet 1153 → Confirmed
1157   Planet 1154 → Confirmed
1158   Planet 1155 → FalsePositive
1159   Planet 1156 → FalsePositive
1160   Planet 1157 → Confirmed
1161   Planet 1158 → FalsePositive
1162   Planet 1159 → FalsePositive
1163   Planet 1160 → Confirmed
1164   Planet 1161 → Confirmed
```

```
1970   Planet 1967 → FalsePositive
1971   Planet 1968 → FalsePositive
1972   Planet 1969 → FalsePositive
1973   Planet 1970 → Confirmed
1974   Planet 1971 → Confirmed
1975   Planet 1972 → FalsePositive
1976   Planet 1973 → Confirmed
1977   Planet 1974 → FalsePositive
1978   Planet 1975 → FalsePositive
1979   Planet 1976 → FalsePositive
1980   Planet 1977 → Confirmed
1981   Planet 1978 → FalsePositive
1982   Planet 1979 → Confirmed
1983   Planet 1980 → Confirmed
1984   Planet 1981 → FalsePositive
1985
```

F. Usage Instructions

- My code just runs as cargo run main, and cargo test for the tests. The only thing out of the ordinary is that the planets are generated in an external .txt file that you have to access, but it should be generated in the same folder.