Samuel Kunkleman

Tutorial:

The purpose of this java project is to discover whether SNP density across SARS-CoV-2 sequences (from the Wuhan reference genome) can predict the length of time the related strain sticks around in the population. Three different models are compared to find the one that seems most predictive. None of them do great yet, but the dataset is fairly small. It's possible that there is no real correlation, though.

Setup:

1. Add jar files in GitHub\javaPractice\finalProject\src\finalProject\weka-3-8-5\ to classpath in eclipse

Data Prep: (done separately in Unix and python)

<u>SNP density counting</u>

Rough code: GitHub\javaPractice\finalProject\src\finalProject\getSNPdensity.sh

Input: vcf output files from artic medaka coronavirus sequencing pipeline

Output: snp counts per kb for all ~30 kbases of Sars-CoV2 genome (one file for each sample)

Description:

- Used vcftools on vcf files to count snps

<u>Concatenation and pairing to IDs</u>

Code: in GitHub\javaPractice\finalProject\src\finalProject\snp_density.ipynb

Input: snp_density csv files, metadata files with sample info

Output: csv with all snp density per kb data and length of occurrence of

Description:

- All inputs were concatenated into csv files and sent to the next step
- Combined info about
    - snp density of each sample &
    - the time that the related coronavirus strain appeared in our region
- This was the input (after being split into test/training sets for the next step

<u>Training/testing sets</u>

Split up dataset (csv) into large training set and small test set

CSVtoARFF: for Loading in CSVs:

Input: csv data file

Output: arrf version of csv

Description: This can convert any csv with header as 1st line to arrf.

Uses for the project:

To convert to .arrf using GitHub\javaPractice\finalProject\src\finalProject\CSVtoARFF.java

- example in main method
- files used in demo are at GitHub\javaPractice\finalProject\
  - snp_density_test.csv → snp_density_test.arrf
  - snp_density_train.csv → snp_density_train.arrf

LinearPredict: for the actual predictions:

Input: (output from CSVtoARFF step 2)

- snp_density_test.arrf
- snp_density_train.arrf

Output:

- Model predictions and comparison of 3 different prediction models

Description:

Weka's classifiers LinearRegression, MultilayerPerceptron, and RandomForest are all trained on the training set and tested on the test set. The terminal output shows the statistical evaluation output provided for each model. The classifier that produced results with lowest overall error is indicated.

Output files (e.g. GitHub\javaPractice\finalProject\LinearRegression_output.csv) exist for each model and show that model's predictions for how long each sample's related coronavirus strain is expected to remain in the population.

None of the models do very well, and that is likely an indication that no real correlation between snp_density (as tested) exists that can predict strain persistence.

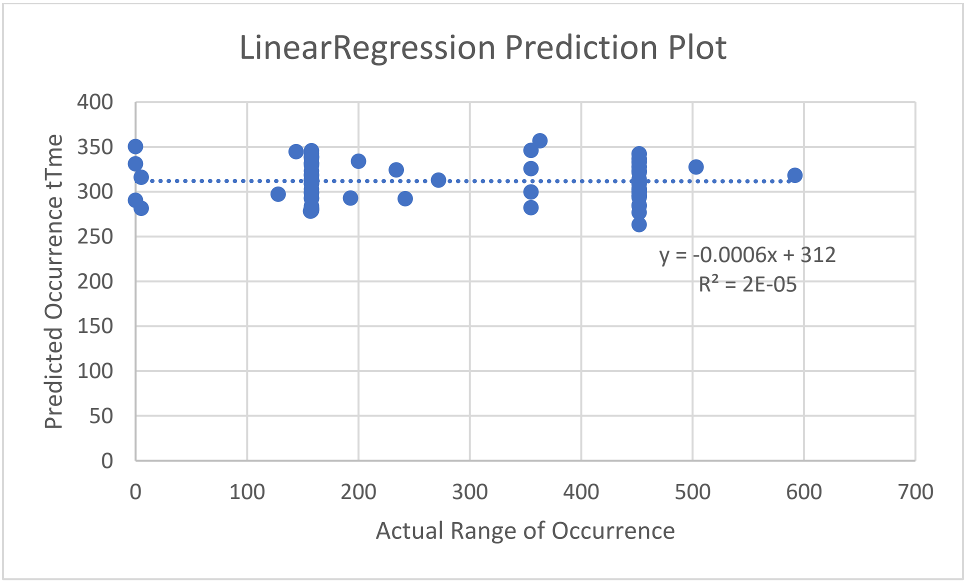
The actual output of LinearPredict looks like this:

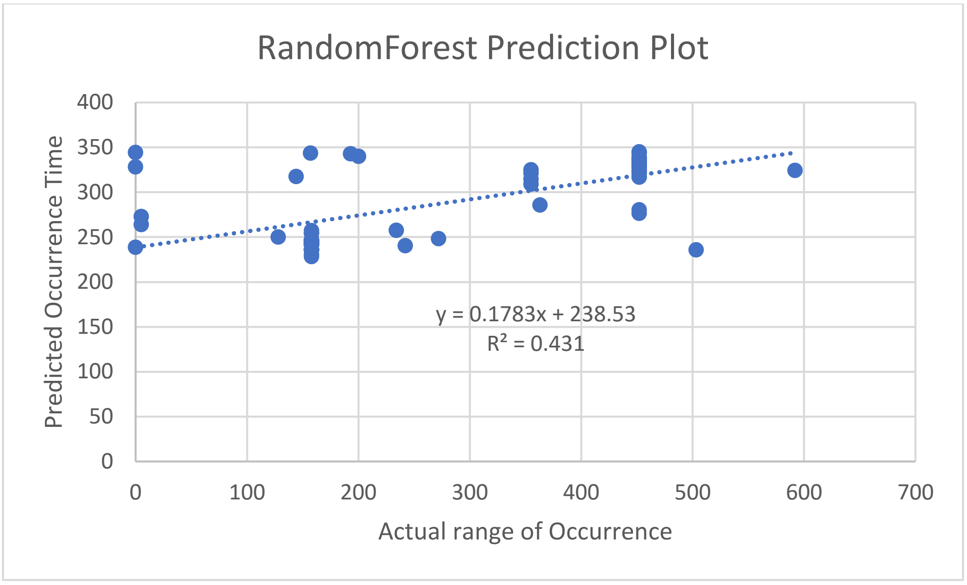
```
Loading data

Creating MultilayerPerceptron model

MultilayerPerceptron evaluation results:
Correlation coefficient                  0.5491
Mean absolute error                    102.7281
Root mean squared error                132.7296
Relative absolute error                 68.4896 %
```

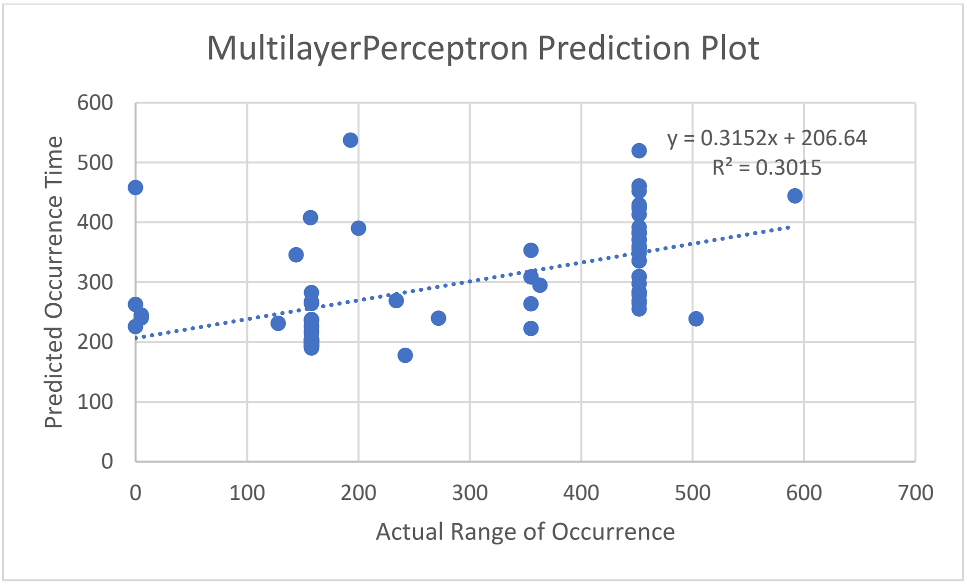
```
Root relative squared error             83.3249 %
Total Number of Instances               74

Creating RandomForest model

RandomForest evaluation results:
Correlation coefficient                  0.6565
Mean absolute error                    118.0245
Root mean squared error                134.2188
Relative absolute error                 78.6879 %
Root relative squared error             84.2598 %
Total Number of Instances               74

Creating LinearRegression model

LinearRegression evaluation results:
Correlation coefficient                 -0.0042
Mean absolute error                    149.3703
Root mean squared error                161.4394
Relative absolute error                 99.5863 %
Root relative squared error            101.3484 %
Total Number of Instances               74

The best classifier was MultilayerPerceptron
```

Graphical depictions:

The following are plots showing the actual values for time each sample's strain persisted in the population versus predicted time ranges for each model.







As seen by the nearly nonexistent slope and R^2 value in the LinearPrediction plot, that classifier had no apparent success with it’s predictions. The other two classifiers at least had some positive relationship between actual and predicted time. But, for both of them, the low R^2 values suggest that the data has too much variance from the trendline to be very predictive. Based on relative absolute error, the MultiLayerPerceptron seems to be the best predictor of this data, but there is more to trusting results that a slightly lower error rate. Based on these poor scores and highly variable output, it seems that none of these classifiers are able to predict the length of strain occurrence in the population based on SNP density.