

Title: Predicting large Interpro accession from small Interpro accessions using DASK



Student: Dilton Thomas

Student number: 436857

Study: DSLS

FINAL ASSIGNMENT

Project1: For the final assignment, I chose project 1

Aim:

The goal of the assignment is to predict large Interpro accession using small Interpro accession. A small Interpro accession is defined as Interpro accession whose feature length(Start-Stop), divided by Sequence length is less than .90. A large Interpro accession is defined as Interpro accession whose feature length(Start-Stop) divided by Sequence length is greater than .90.

Exploratory Data Analysis:

Here, I analysed the distribution of feature length (Start -Stop) of small and large Interpro accessions using a histogram.

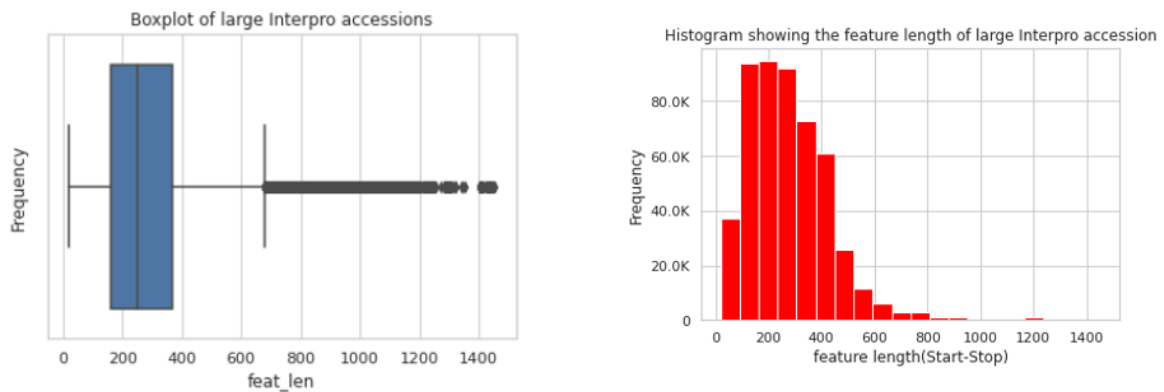


Figure 1 Boxplot and Histogram of large Interpro accessions

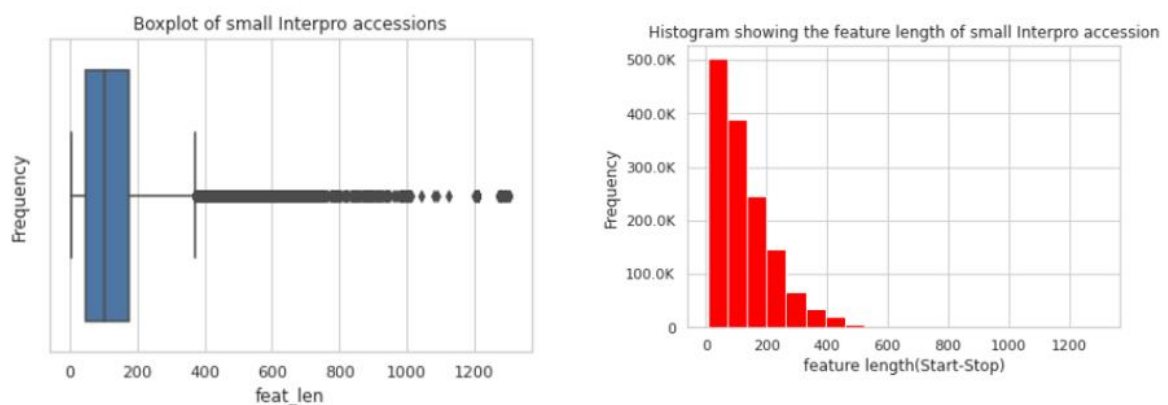


Figure 2 Boxplot and Histogram of small Interpro accessions

Data Pre-processing:

For data pre-processing, columns were renamed, removed missing Interpro accession and duplicates from the dataset.

Methods:

Pyspark was used to get the histogram of the large and small Interpro accession. The whole data pre-processing and the making of machine learning was implemented using Dask. I only kept proteins that have at least one large and small Interpro accession. If a protein has multiple large Interpro accession, the largest Interpro accession out of all large Interpro accession is filtered out. Some proteins showed multiple small Interpro accession with the same name. For these cases, the count of small Interpro accessions is calculated.

For machine learning, I chose Randomforest classifier because it's an ensemble method and works with large datasets. 80 per cent of the data was used for model training and the rest for testing purposes. Furthermore, One hot Encoding was done on classes(largest Interpro accessions) to make it suitable for the fit method of sklearn.

Results:

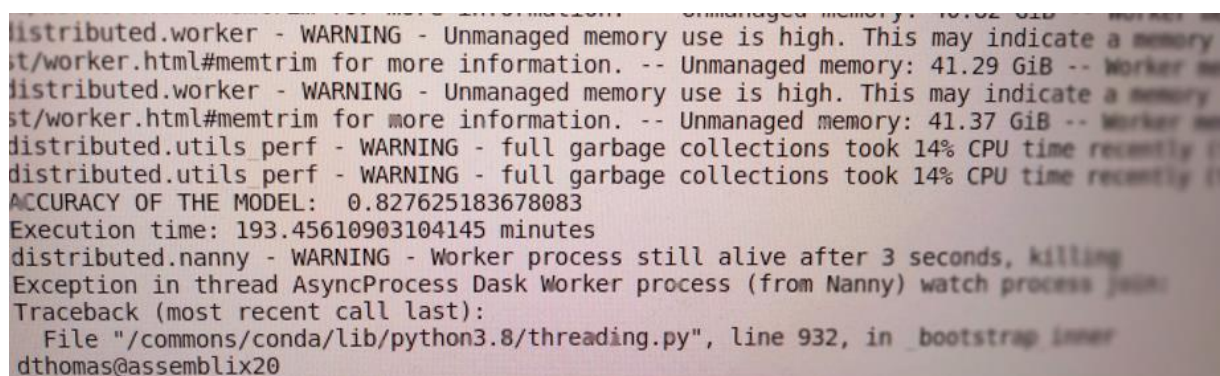
I trained and tested Random Forests on 10000,100000 and 1 million lines of data from bacilli.tsv file and I got 24, 66 and 82 per cent accuracy for the model on test data. The accuracy of Random forest is 82 per cent when the first 1 million lines of the data set were used. The final model was exported to a pickle file and the training was saved to csv file. The files containing the model in pickle format and training data are saved in the directory .

Random forest as pickle format is saved in:

```
'/students/2021-2022/master/Dilton_DSLS/randforest_model1.pkl'
```

Training data as csv format is saved in:

```
'/students/2021-2022/master/Dilton_DSLS/ X_train.csv'
```



```
distributed.worker - WARNING - Unmanaged memory use is high. This may indicate a memory
st/worker.html#memtrim for more information. -- Unmanaged memory: 41.29 GiB -- Worker m
distributed.worker - WARNING - Unmanaged memory use is high. This may indicate a memory
st/worker.html#memtrim for more information. -- Unmanaged memory: 41.37 GiB -- Worker m
distributed.utils_perf - WARNING - full garbage collections took 14% CPU time recently
distributed.utils_perf - WARNING - full garbage collections took 14% CPU time recently
ACCURACY OF THE MODEL: 0.827625183678083
Execution time: 193.45610903104145 minutes
distributed.nanny - WARNING - Worker process still alive after 3 seconds, killing
Exception in thread AsyncProcess Dask Worker process (from Nanny) watch process join:
Traceback (most recent call last):
  File "/commons/conda/lib/python3.8/threading.py", line 932, in _bootstrap inner
dthomas@assemblix20
```

Figure 3. Figure showing the accuracy and time taken by Random Forest

Improvements for the future:

- 1) Incremental training of large datasets using Dask ML incremental estimator. However, this sometimes doesn't support all ML algorithms.
- 2) Converting the training Dask dataframe to a sparse matrix. This will significantly reduce the amount of memory for training data.
- 3) Implementing XG boost for the current dataset and tuning hyper parameters of the current Random Forest model.
- 4) Finally, for this assignment combination of Pyspark and Dask would work the best. All the data processing up to the clean dataframe for machine learning can be achieved faster in Pyspark. Then the Pyspark dataframe can be converted to Pandas dataframe and then to Dask dataframe for machine learning.