Revanth Korrapolu

Documentation

**1. Download two datasets from the Uniprot database (**[**https://www.uniprot.org**](https://www.uniprot.org/)**), (a) all eukaryotic protein sequences and (b) all prokaryotic sequences**

to download all eukaryotic protein sequences:

uniparc proteome:(taxonomy:"Eukaryota [2759]")

to download all prokaryotic protein sequences:

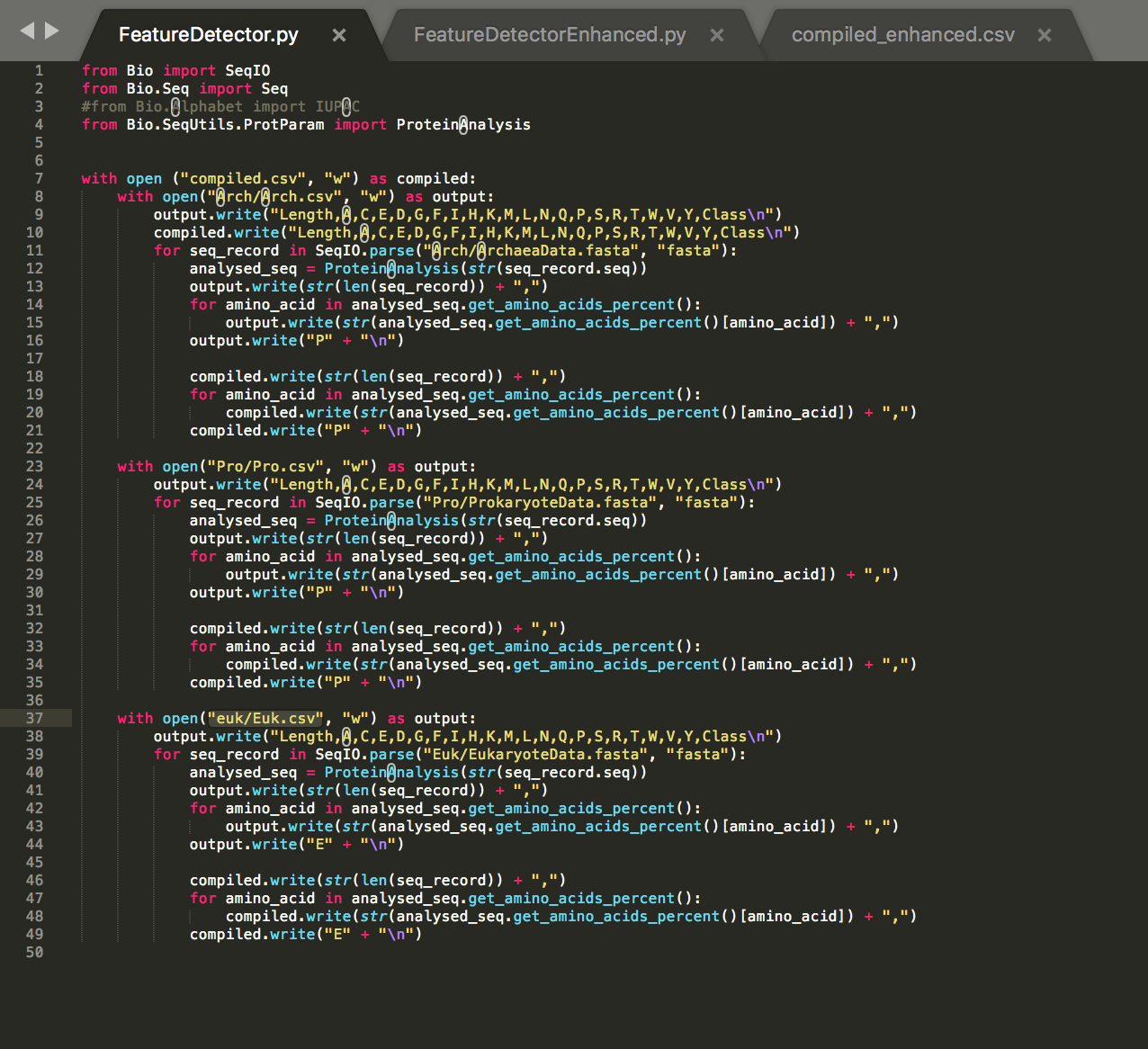
uniparc proteome:(taxonomy:"Eukaryota [2759]")

taxonomy:"Archaea [2157]"

taxonomy:"Bacteria [2]"

**2. Compute two features for each of those sequences: (1) length and (2) amino acid composition**

Created a simple script using biopython to parse for amino acid composition and sequence length. (See FeatureDetector.py)

****

**3. Create a training set containing all features for both sets of sequences and a class label (0: prokaryotic, 1: eukaryotic)**

Yes: <https://github.com/RevanthK/ML-Protein-Classifier>

The prokaryotic training set is in Pro/Pro.csv

The Archaea training set is in Arch/Arch.csv

The eukaryotic training set is in Euk/Euk.csv

There is also a compiled training with all classes in compiled.csv.

**4. User weka to load this training set and**

1. **create a pipeline which uses 10-fold cross validation to train a model (appropriate for the input features)**

yes

**b) the pipeline should use a balancing filter to make sure the training dataset has an equal amount of class instances**

Using supervised class balancer filter

**c) report model performance, d) apply parameter optimization**

The two most models which I ran include the multilayer perceptron and logistical regression. They correctly classified on 83% and 81% of the instances. Overall, the perceptron proved to be the most accurate and however the model took very long to train (387 seconds). I would opt for the logistical regression model because it provided a negligible difference in accuracy but required significantly less time to train (6 seconds).

The condensed Weka log outputs from the two models are listed below:

=== Run information ===

Scheme: weka.classifiers.functions.**MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Time taken to build model: 387.7 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 127471.8658 83.9492 %

Incorrectly Classified Instances 24372.1342 16.0508 %

Kappa statistic 0.679

Mean absolute error 0.2094

Root mean squared error 0.3472

Relative absolute error 41.8857 %

Root relative squared error 69.4345 %

Total Number of Instances 151844

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.824 0.145 0.850 0.824 0.837 0.679 0.906 0.895 P

0.855 0.176 0.829 0.855 0.842 0.679 0.906 0.895 E

Weighted Avg. 0.839 0.161 0.840 0.839 0.839 0.679 0.906 0.895

=== Confusion Matrix ===

a b <-- classified as

62582.5 13339.5 | a = P

11032.63 64889.37 | b = E

**################################################################**

Scheme: weka.classifiers.functions.**Logistic** -R 1.0E-8 -M -1 -num-decimal-places 4

Relation: compiled-weka.filters.supervised.instance.ClassBalancer-num-intervals10

Time taken to build model: 6.25 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 123600.3662 81.3996 %

Incorrectly Classified Instances 28243.6338 18.6004 %

Kappa statistic 0.628

Mean absolute error 0.2845

Root mean squared error 0.3707

Relative absolute error 56.9097 %

Root relative squared error 74.1491 %

Total Number of Instances 151844

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.833 0.205 0.802 0.833 0.818 0.628 0.882 0.855 P

0.795 0.167 0.827 0.795 0.810 0.628 0.882 0.875 E

Weighted Avg. 0.814 0.186 0.814 0.814 0.814 0.628 0.882 0.865

=== Confusion Matrix ===

a b <-- classified as

63267.8 12654.2 | a = P

15589.44 60332.56 | b = E