**DNA Sequencing classification in Deep Machine Learning**

**Introduction**

A genome is a complete collection of DNA in an organism. All living species possess a genome, but they differ considerably in size. As a data-driven science, genomics extensively utilizes machine learning to capture dependencies in data and infer new biological hypotheses. Nonetheless, the ability to extract new insights from the exponentially increasing volume of genomics data requires more powerful machine learning models. By efficiently leveraging large data sets, deep learning has reconstructed fields such as computer vision and natural language processing. It has become the method of preference for many genomics modeling tasks, including predicting the influence of genetic variation on gene regulatory mechanisms such as DNA receptiveness and splicing. So here, I will understand DNA structure and how machine learning can be used to work with DNA sequence data.

In this project, I will apply a classification model that can predict a gene's function based on the DNA sequence of the coding sequence alone. The whole project uses human bases, dog bases, and orangutan bases. After programming, a total of 6,720 pairs are used to convert text into characters and numbers for classification.

**Material and Method**

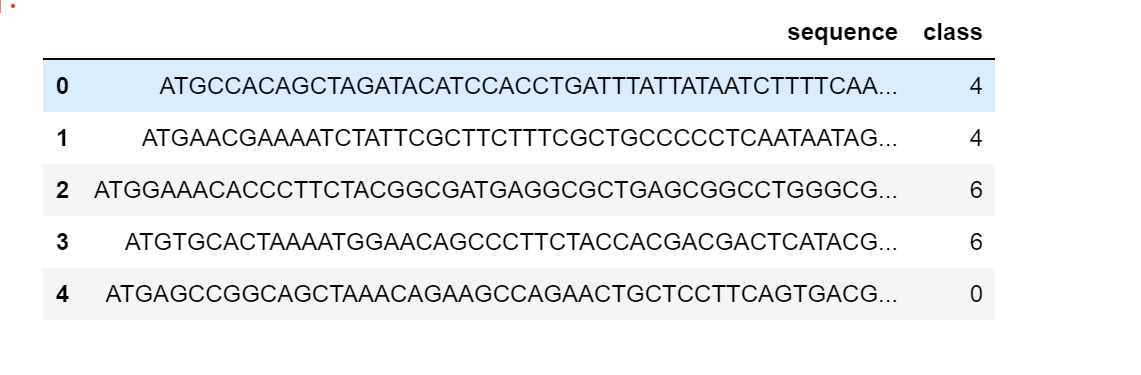
I use Python to perform deep machine learning DNA classification of humans, dogs, and chimp with a total of 60k bases. I use k-mer counting to classify and count them. Finally, I use training and test data and Naive Bayes to view model indicators, see the accuracy and do visualize results by Python package. Of course, when dividing bases and pairing, some biological special methods are needed. Treating DNA sequence as a "language", otherwise known as k-mer counting.

A challenge that remains is that none of these above methods results in vectors of uniform length, and that is a requirement for feeding data to a classification or regression algorithm. So with the above methods you have to resort to things like truncating sequences or padding with "n" or "0" to get vectors of uniform length. The method I use here is simple and easy. I first take the long biological sequence and break it down into k-mer length overlapping “words”. For example, if I use "words" of length 6 (hexamers), “ATGCATGCA” becomes: ‘ATGCAT’, ‘TGCATG’, ‘GCATGC’, ‘CATGCA’. Hence our example sequence is broken down into 4 hexamer words.

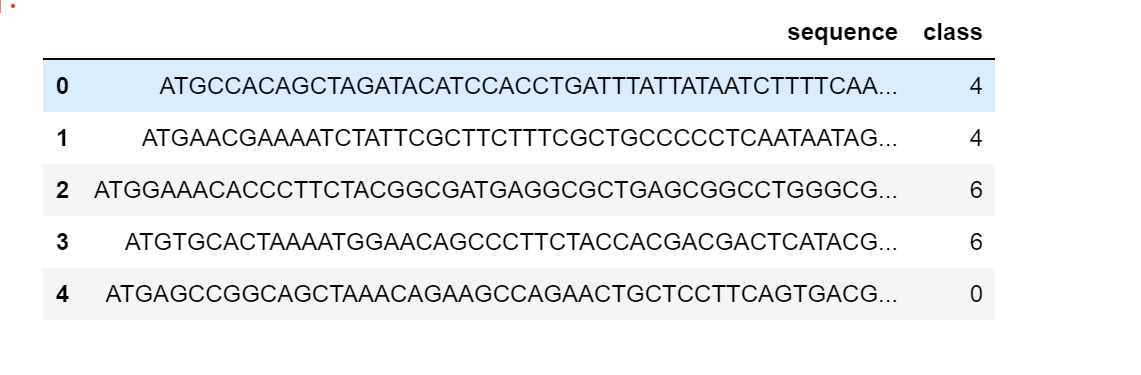
Here I am using hexamer “words” but that is arbitrary and word length can be tuned to suit the particular situation. The word length and amount of overlap need to be determined empirically for any given application. In genomics, we refer to these types of manipulations as "k-mer counting", or counting the occurances of each possible k-mer sequence. There are specialized tools for this, but the Python natural language processing tools make it supe easy.

**Discussion**

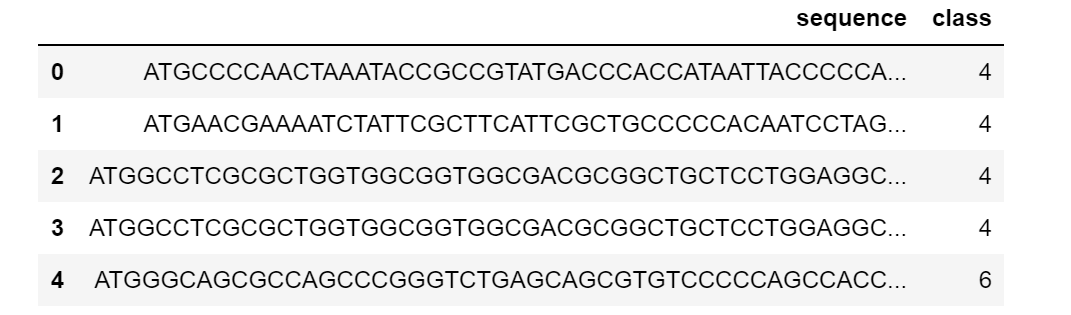
Check DNA seq data:



Human DNA seq

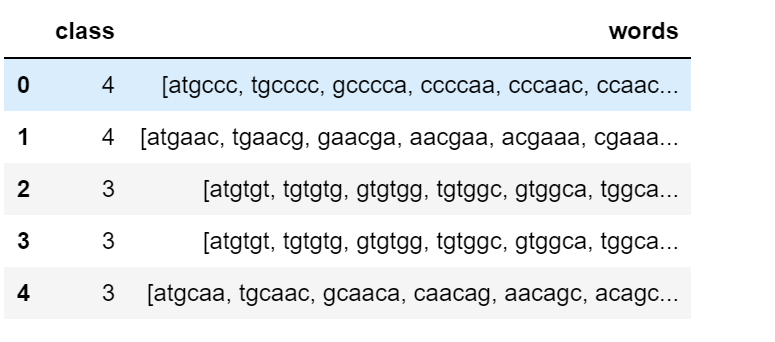


Chimp DNA seq

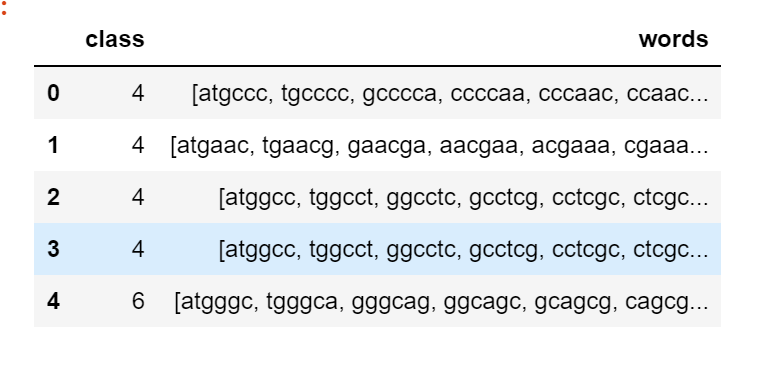


Dog DNA seq

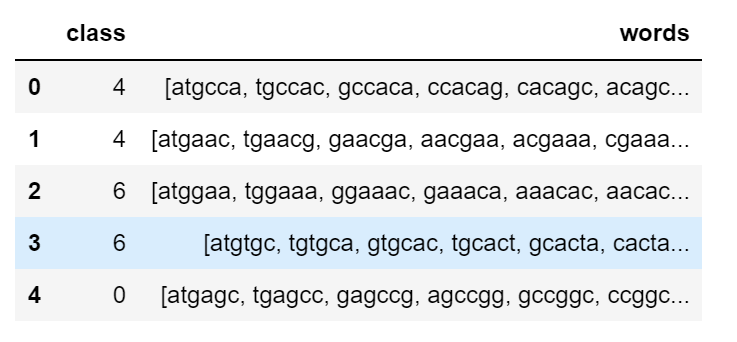
### Our coding sequence data is changed to lowercase, split up into all possible k-mer words of length 6 and ready for the next step. Let's take a look：



Human Data

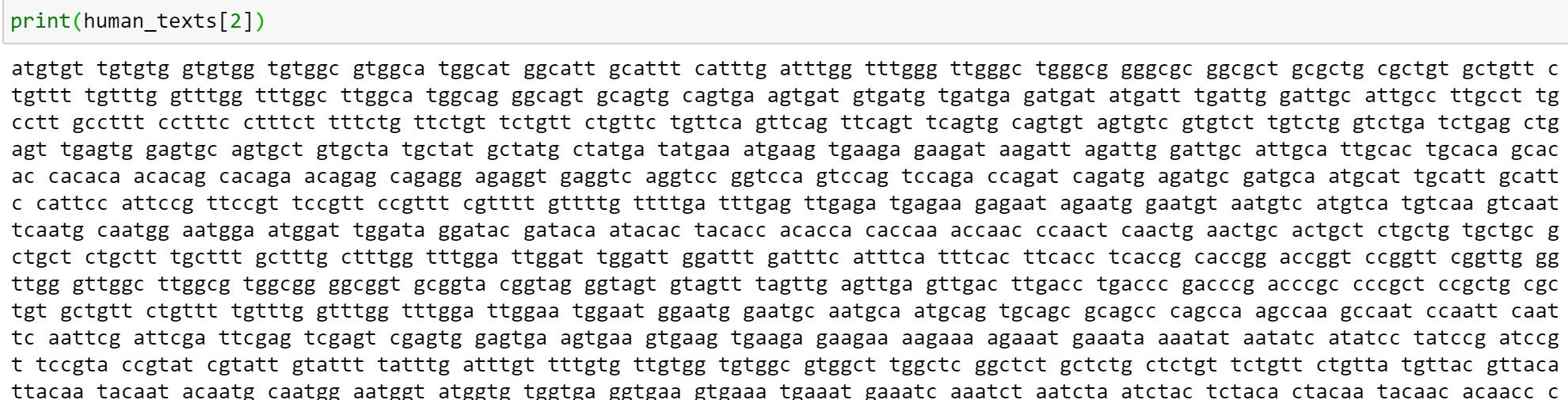


Chimp Data

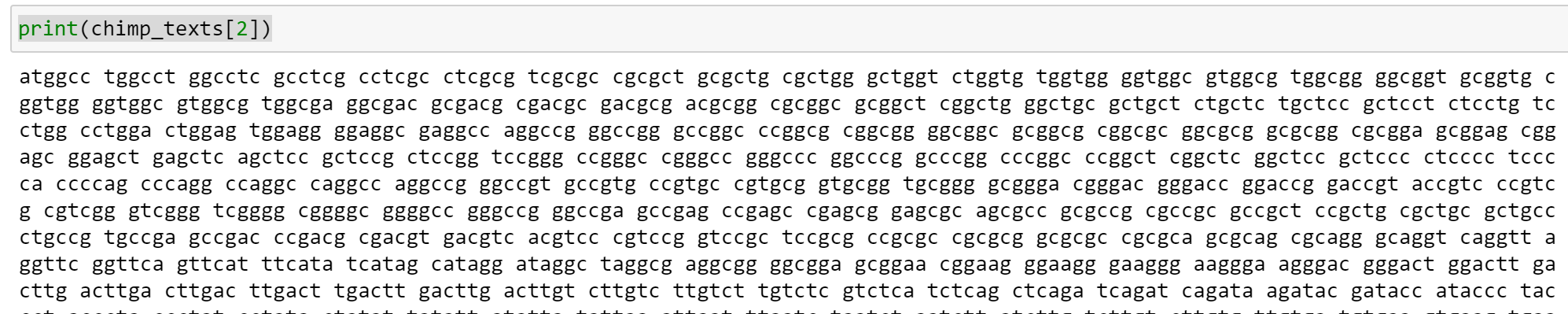


Dog Data

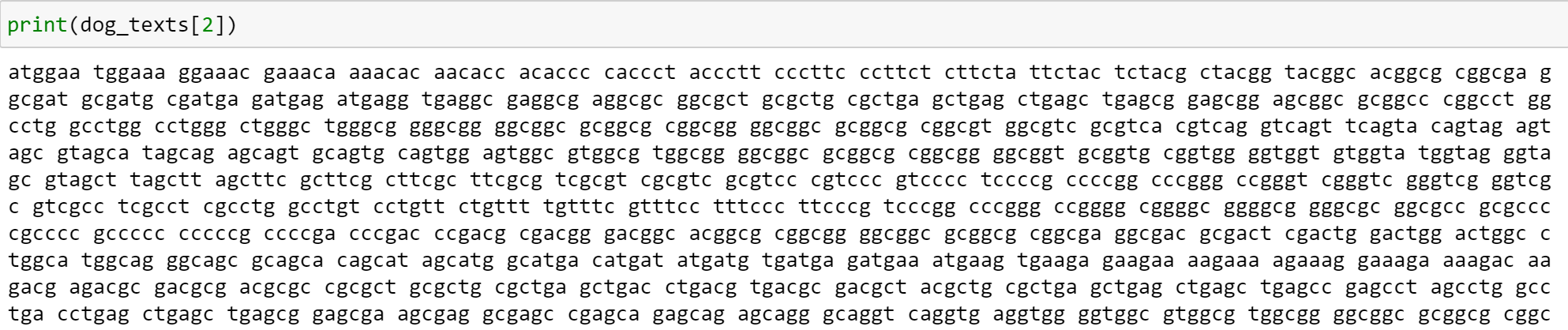
Since we are going to use scikit-learn natural language processing tools to do the k-mer counting, we need to now convert the lists of k-mers for each gene into string sentences of words that the count vectorizer can use. We can also make a y variable to hold the class labels.



Human DNA bases



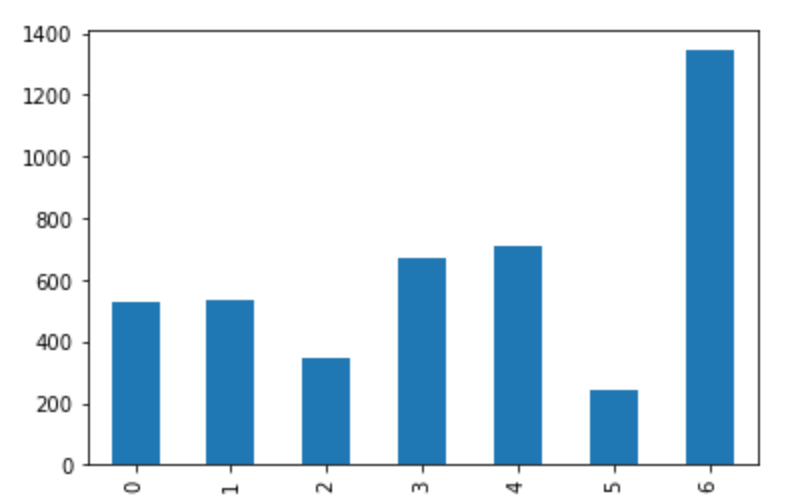
Chimp DNA bases



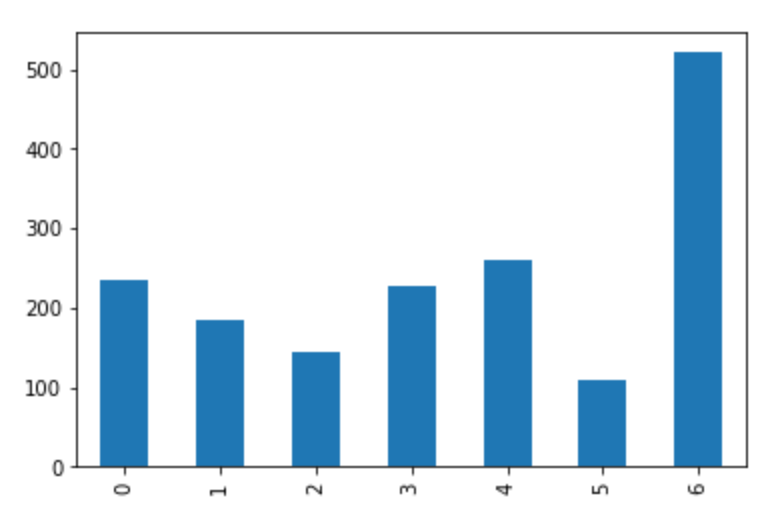
Dog DNA bases

**Visualize results**

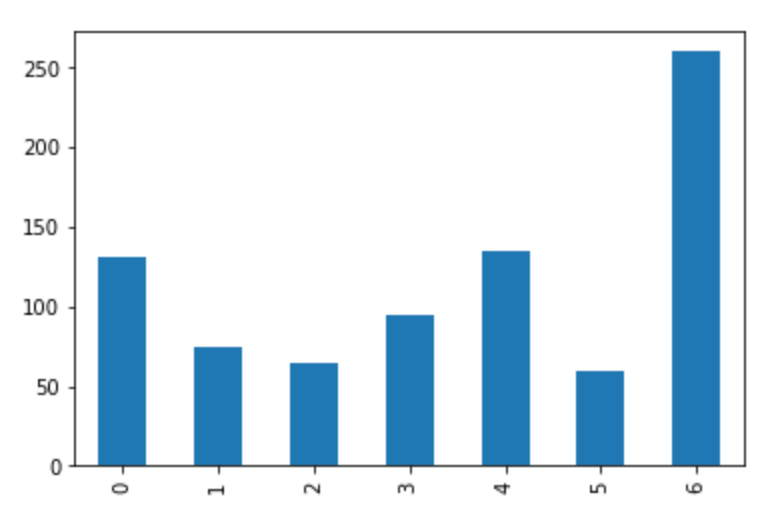
According to the string count, we can see the distribution of the number of bases in each group through the graph



Human Data



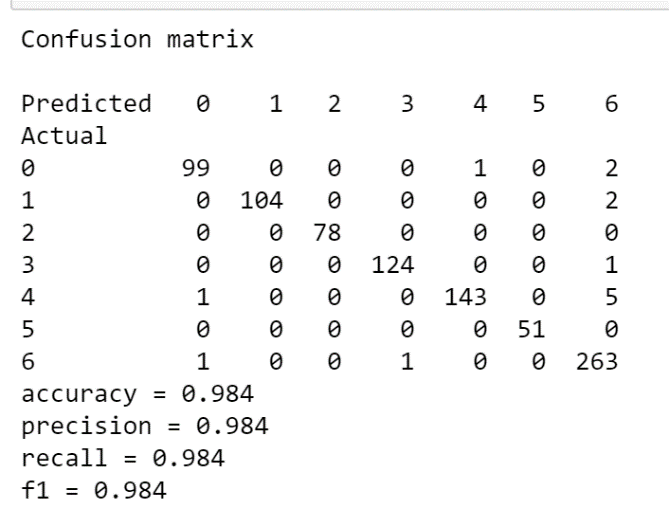
Chimp Data



Dog data

#### **Model and Training Data**

#### A multinomial naive Bayes classifier will be created. I previously did some parameter tuning and found the ngram size of 4 (reflected in the Countvectorizer() instance) and a model alpha of 0.1 did the best. So let's look at some model performce metrics like the confusion matrix, accuracy, precision, recall and f1 score. We are getting really good results on our unseen data, so it looks like our model did not overfit to the training data. In a real project I would go back and sample many more train test splits since we have a relatively small data set.



**Conclusion**

In this project, I explore the world of bioinformatics by using K-mers counting support classify DNA sequences. I classify the bases, viewed, counted, and processed the DNA of different groups through the machine language Python. Then, the research process and results are better displayed through the visualization results. Finally, the accuracy of the classification is judged by the Multinomial Naive Bayes Classifier model. The accuracy is good, and the training data and results are perfect. This will inspire my future application of combining data learning into biology.

**Reference**

Deep Learning Architectures for DNA Sequence Classification, Giosué Lo BoscoMattia Antonino Di Gangi, 2017

A new feature selection methodology for K-mers representation of DNA sequences. In: Serio, C., Liò, P., Nonis, A., Tagliaferri, R. (eds.) CIBB 2014. Heidelberg (2015)

Base pair in National Human Genome Research Institute, 2020