Computing for Bioinformatics
Group Project
Student's particulars

Names	Registration#
Muhammad Shah Zaib	402305
Hamna Imran	402029
Maria Ilyas Malik	402080

Contents

Problem Statement:

One level of genome organization is the grouping of genes into several gene families. Gene families are groups of related genes that share a common ancestor. Members of gene families may be paralogs or orthologs. **Gene paralogs** are genes with similar sequences from within the same species while **gene orthologs** are genes with similar sequences in different species. Gene families are highly variable in size, sequence diversity, and arrangement.

Objective:

Build a classification model that is trained on the human DNA sequence and can predict a gene family based on the DNA sequence of the coding sequence. To test the model, we will use the DNA sequence of humans, dogs, and chimpanzees and compare the accuracies.

Scope:

Machine learning (ML) has been instrumental in optimal decision-making through relevant historical data, including the domain of bioinformatics. In bioinformatics classification of natural genes and the genes that are infected by a disease called invalid genes is a very complex task. To find the applicability of a fresh protein through genomic research, DNA sequences need to be classified. The current work identifies classes of DNA sequences using a machine-learning algorithm.

These classes are basically dependent on the sequence of nucleotides. With a fractional mutation in sequence, there is a corresponding change in the class. Each numeric instance representing a class is linked to a gene family including G protein-coupled receptors, tyrosine kinase, synthase, etc. In this project, we applied the classification algorithm on three types of datasets to identify which gene class they belong to. They converted sequences into substrings with a defined length. That ‘k value’ defines the length of the substring which is one of the ways to analyze the sequence.

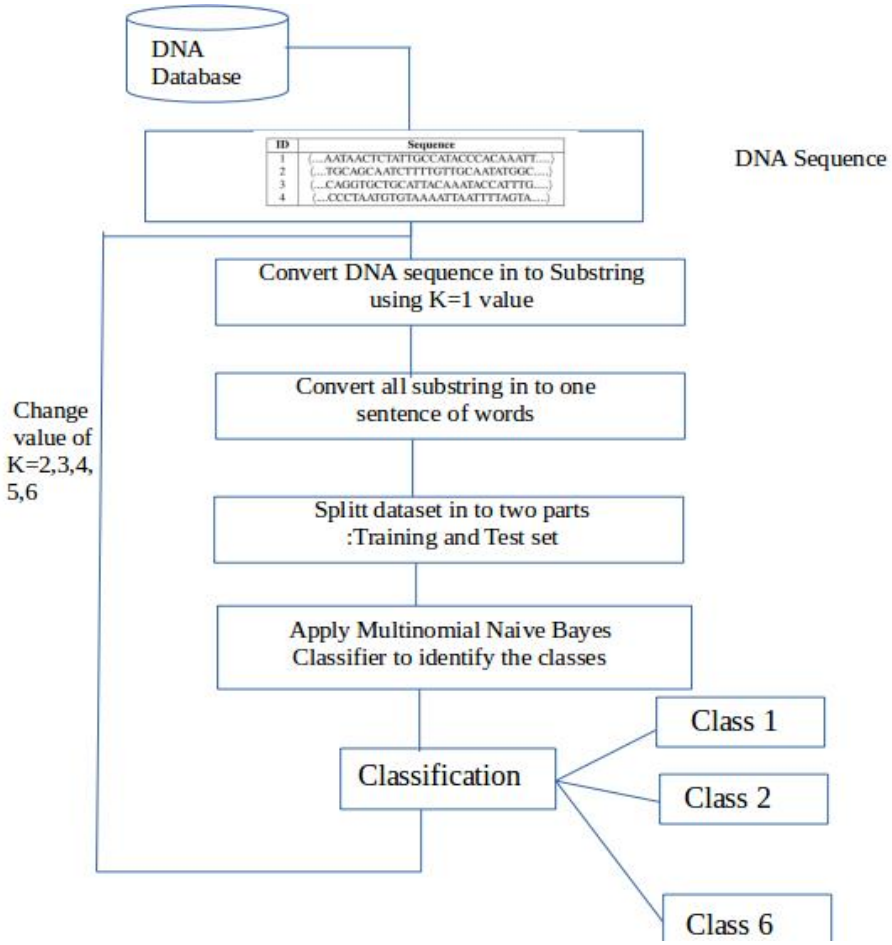
- We just worked on the DNA sequencing problem but there are several other issues also where we can continue our work like RNA sequencing and protein sequence.
- Deep learning models can also be developed to classify diseases like COVID, MERS, SARS, dengue, hepatitis, influenza, etc.
- It will help to see the relationship among them whether they are related to each other and help in identifying the ancestral history.
- We can also implement such classification models on patients’ DNA Sequences to help them develop personalized medicines. It would be cost and time efficient

Project Execution:

The processing is done in distinct phases as described below:

- **STEP 1**: First convert DNA sequence strings into k-mer words, default size = 6. If we choose size=4, then it breaks the whole sequence of nucleotides into 4 nucleotides. First four characters are taken, next time it skips the first character and takes next four nucleotides.
- **STEP 2**: After that we need to convert all substrings in list into one sentence of words so that we can manage it easily with one variable.
- **STEP 3**: Splitting the dataset into two parts i.e., training set and test set. Here we divide into an 80:20 ratio means 80% data we use as training data and 20% as testing data.
- **STEP 4**: Now apply the Multinomial Naive Bayes Classifier to identify the classes. Using grid search we can fix the alpha value.
- **STEP 5**: Now check whether kmer counting is working on gene sequence or not, checked through the confusion matrix.

Flowchart of modules implemented:



Code and Output:

Importing the libraries that are used in the classification modeling

```
#importing libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
```

we apply the Multinomial algorithm on a variety of datasets i.e., Chimpanzee dataset and Dog dataset and human Dataset, and evaluate the effect of k value on these types of datasets. We used python to implement that. First, we apply multinomial algorithm on dataset to classify the gene sequence into its classes as a classifier.

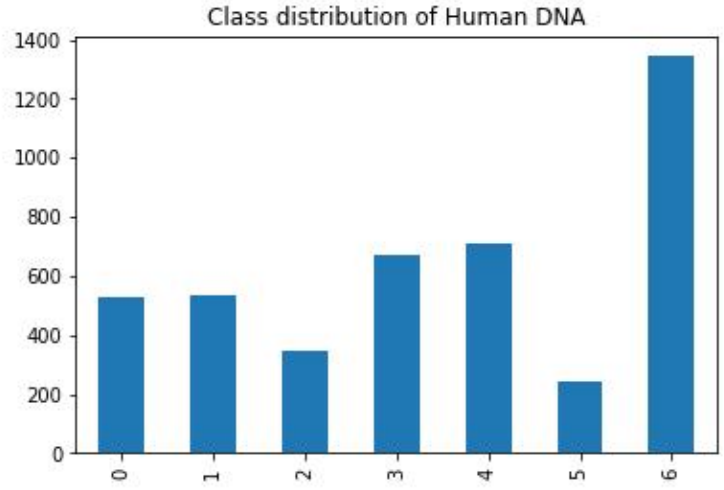
For human dataset

```
#Load human DNA data
human_dna = pd.read_table('/content/human.txt') # Read general delimited file into DataFrame
human_dna.head()
```

	sequence	class
0	ATGCCCCAACTAAATACTACCGTATGGCCCACCATAATTACCCCCA...	4
1	ATGAACGAAAATCTGTTCGCTTCATTCATTGCCCCCACAATCCTAG...	4
2	ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG...	3
3	ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG...	3
4	ATGCAACAGCATTTTGAATTTGAATACCAGACCAAAGTGGATGGTG...	3

```
human_dna['class'].value_counts().sort_index().plot.bar()
plt.title("Class distribution of Human DNA")
```

```
Text(0.5, 1.0, 'Class distribution of Human DNA')
```



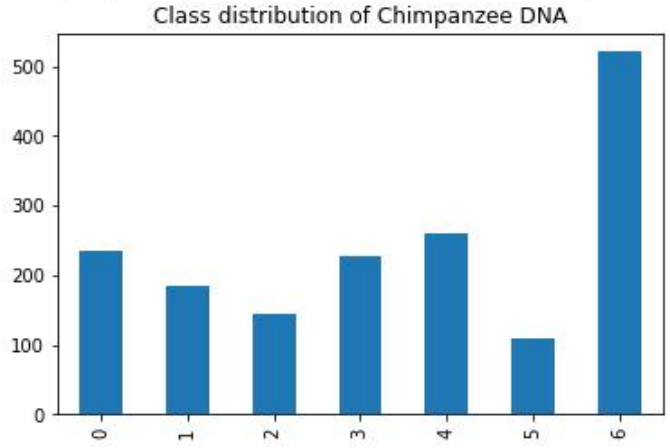
For chimpanzee dataset

```
#Load Chimpanzee DNA data
chimp_dna = pd.read_table('/content/chimpanzee.txt')
chimp_dna.head()
```

	sequence	class
0	ATGCCCCAACTAAATACCGCCGTATGACCCACCATAATTACCCCCA...	4
1	ATGAACGAAAATCTATTCGCTTCATTCGCTGCCCCCACAATCCTAG...	4
2	ATGGCCTCGCGCTGGTGGCGGTGGCGACGCGGCTGCTCCTGGAGGC...	4
3	ATGGCCTCGCGCTGGTGGCGGTGGCGACGCGGCTGCTCCTGGAGGC...	4
4	ATGGGCAGCGCCAGCCCGGGTCTGAGCAGCGTGTCCCCCAGCCACC...	6

```
chimp_dna['class'].value_counts().sort_index().plot.bar()
plt.title("Class distribution of Chimpanzee DNA")
```

```
Text(0.5, 1.0, 'Class distribution of Chimpanzee DNA')
```



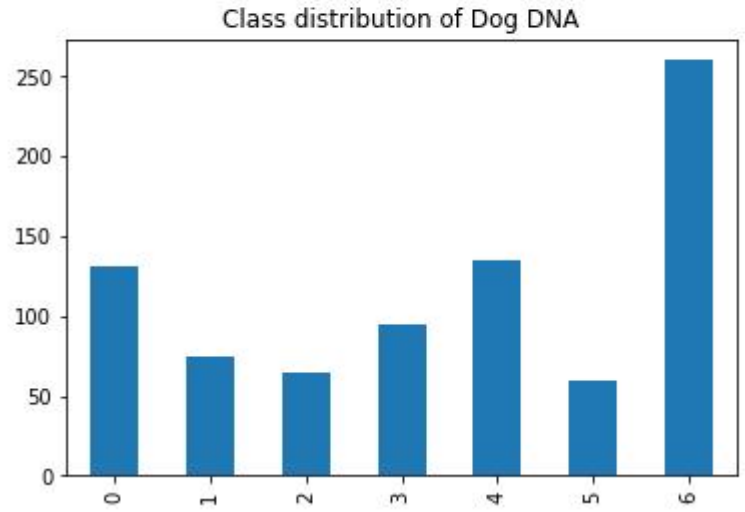
For dog dataset

```
#Load Dog DNA data
dog_dna = pd.read_table('/content/dog.txt')
dog_dna.head()
```

	sequence	class
0	ATGCCACAGCTAGATACATCCACCTGATTTATTATAATCTTTTCAA...	4
1	ATGAACGAAAATCTATTCGCTTCTTTCGCTGCCCCCTCAATAATAG...	4
2	ATGGAAACACCCTTCTACGGCGATGAGGCGCTGAGCGGCCTGGGCG...	6
3	ATGTGCACTAAAATGGAACAGCCCTTCTACCACGACGACTCATACG...	6
4	ATGAGCCGGCAGCTAAACAGAAGCCAGAACTGCTCCTTCAGTGACG...	0

```
dog_dna['class'].value_counts().sort_index().plot.bar()
plt.title("Class distribution of Dog DNA")
```

```
Text(0.5, 1.0, 'Class distribution of Dog DNA')
```



Now we have all our data loaded, the next step is to convert a sequence of characters into k-mer words, default **size = 6 (hexamers)**. The function **Kmers_funct()** will collect all possible overlapping k-mers of a specified length from any sequence string.

```
def Kmers_funct(seq, size=6):
    return [seq[x:x+size].lower() for x in range(len(seq) - size + 1)]

#convert our training data sequences into short overlapping k-mers of length 6.
#Lets do that for each species of data we have using our Kmers_funct function.

human_dna['words'] = human_dna.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
human_dna = human_dna.drop('sequence', axis=1)

chimp_dna['words'] = chimp_dna.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
chimp_dna = chimp_dna.drop('sequence', axis=1)

dog_dna['words'] = dog_dna.apply(lambda x: Kmers_funct(x['sequence']), axis=1)
dog_dna = dog_dna.drop('sequence', axis=1)
```

The DNA sequence is changed to lowercase, divided into all possible k-mer words of length 6, and ready for the next step.

```
human_dna.head()
```

	class	words
0	4	[atgccc, tgcccc, gcccca, ccccaa, cccaac, ccaac...
1	4	[atgaac, tgaacg, gaacga, aacgaa, acgaaa, cgaaa...
2	3	[atgtgt, tgtgtg, gtgtgg, tgtggc, gtggca, tggca...
3	3	[atgtgt, tgtgtg, gtgtgg, tgtggc, gtggca, tggca...
4	3	[atgcaa, tgcaac, gcaaca, caacag, aacagc, acagc...

We need to now convert the lists of k-mers for each gene into string sentences of words that can be used to create the Bag of Words model. We will make a target variable y to hold the class labels.

```
human_texts = list(human_dna['words'])
for item in range(len(human_texts)):
    human_texts[item] = ' '.join(human_texts[item])
#separate labels
y_human = human_dna.iloc[:, 0].values # y_human for human_dna
```

Now let's do the same for chimp and dog.

```
chimp_texts = list(chimp_dna['words'])
for item in range(len(chimp_texts)):
    chimp_texts[item] = ' '.join(chimp_texts[item])
#separate labels
y_chim = chimp_dna.iloc[:, 0].values # y_chim for chimp_dna

dog_texts = list(dog_dna['words'])
for item in range(len(dog_texts)):
    dog_texts[item] = ' '.join(dog_texts[item])
#separate labels
y_dog = dog_dna.iloc[:, 0].values  # y_dog for dog_dna
```

```
y_human
```

```
array([4, 4, 3, ..., 6, 6, 6])
```

So, the target variable contains an array of class values.

Creating the Bag of Words model using CountVectorizer(). This is equivalent to k-mer counting. The n-gram size of 4 was previously determined by testing.

Convert our k-mer words into uniform length numerical vectors that represent counts for every k-mer in the vocabulary:

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(ngram_range=(4,4)) #The n-gram size of 4 is previously determined by testing
X = cv.fit_transform(human_texts)
X_chimp = cv.transform(chimp_texts)
X_dog = cv.transform(dog_texts)
```

```
print(X.shape)
print(X_chimp.shape)
print(X_dog.shape)
```

```
(4380, 232414)
(1682, 232414)
(820, 232414)
```

So, for humans we have **4380** genes converted into uniform length feature vectors of 4-gram k-mer(length 6) counts. For chimp and dog, we have the same number of features with **1682** and **820** genes respectively.

So now that we know how to transform our DNA sequences into uniform length numerical vectors in the form of k-mer counts and ngrams, we can now go ahead and build a classification model that can predict the DNA sequence function based only on the sequence itself.

Here we will use the human data to train the model, holding out 20% of the human data to test the model. Then we can challenge the model's generalizability by trying to predict sequence function in other species (the chimpanzee and dog).

Next, train/test split human dataset and build a simple multinomial naive Bayes classifier.

You might want to do some parameter tuning and build a model with different ngram sizes, here we will go ahead with a ngrams size of 4 and a model alpha of 0.1.

```
# Splitting the human dataset into the training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y_human,
                                                    test_size = 0.20,
                                                    random_state=42)
```

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB(alpha=0.1)
classifier.fit(X_train, y_train)
```

```
MultinomialNB(alpha=0.1)
```

```
#Now let's make predictions on the human hold out test set and see how it performes on unseen data
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print("Confusion matrix for predictions on human test DNA sequence\n")
print(pd.crosstab(pd.Series(y_test, name='Actual'), pd.Series(y_pred, name='Predicted')))
def get_metrics(y_test, y_predicted):
    accuracy = accuracy_score(y_test, y_predicted)
    precision = precision_score(y_test, y_predicted, average='weighted')
    recall = recall_score(y_test, y_predicted, average='weighted')
    f1 = f1_score(y_test, y_predicted, average='weighted')
    return accuracy, precision, recall, f1
accuracy, precision, recall, f1 = get_metrics(y_test, y_pred)
print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))
```

```
Confusion matrix for predictions on human test DNA sequence

Predicted   0    1   2    3    4   5    6
Actual
0          99    0   0    0    1   0    2
1           0  104   0    0    0   0    2
2           0    0  78    0    0   0    0
3           0    0   0  124    0   0    1
4           1    0   0    0  143   0    5
5           0    0   0    0    0  51    0
6           1    0   0    1    0   0  263
accuracy = 0.984
precision = 0.984
recall = 0.984
f1 = 0.984
```

```
# Predicting the chimp, dog sequences
y_pred_chimp = classifier.predict(X_chimp)
```

```
# performance on chimpanzee genes
print("Confusion matrix for predictions on Chimpanzee test DNA sequence\n")
print(pd.crosstab(pd.Series(y_chim, name='Actual'), pd.Series(y_pred_chimp, name='Predicted')))
accuracy, precision, recall, f1 = get_metrics(y_chim, y_pred_chimp)
print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))
```

```
Confusion matrix for predictions on Chimpanzee test DNA sequence

Predicted    0    1    2    3    4    5    6
Actual
0          232    0    0    0    0    0    2
1            0  184    0    0    0    0    1
2            0    0  144    0    0    0    0
3            0    0    0  227    0    0    1
4            2    0    0    0  254    0    5
5            0    0    0    0    0  109    0
6            0    0    0    0    0    0  521
accuracy = 0.993
precision = 0.994
recall = 0.993
f1 = 0.993
```

```
y_pred_dog = classifier.predict(X_dog)
```

```
# performance on dog genes
print("Confusion matrix for predictions on Dog test DNA sequence\n")
print(pd.crosstab(pd.Series(y_dog, name='Actual'), pd.Series(y_pred_dog, name='Predicted')))
accuracy, precision, recall, f1 = get_metrics(y_dog, y_pred_dog)
print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))
```

```
Confusion matrix for predictions on Dog test DNA sequence

Predicted    0   1   2   3    4   5    6
Actual
0          127   0   0   0    0   0    4
1            0  63   0   0    1   0   11
2            0   0  49   0    1   0   14
3            1   0   0  81    2   0   11
4            4   0   0   1  126   0    4
5            4   0   0   0    1  53    2
6            0   0   0   0    0   0  260
accuracy = 0.926
precision = 0.934
recall = 0.926
f1 = 0.925
```

The model seems to produce good results on human data. It also does on Chimpanzee which is because the Chimpanzee and humans share the same genetic hierarchy. The performance of the dog is not quite as good which is because the dog is more diverging from humans than the chimpanzee.

Limitations:

- Large data sets are the key to machine learning. At present, the magnitude of most biological data sets is still too small to meet the requirements of machine learning algorithms. Although the total amount of biological data is huge and increasing day by day, the collection of data comes from different platforms. Due to the differences in technology and biology itself, it is very difficult to integrate different data sets.
- Due to the differences in biological data itself, machine learning models trained on one data set may not be well generalized to other data sets. If the new data is significantly different from the training data, the analysis results of the machine learning model are likely to be false.
- The black-box nature of machine learning models brings new challenges to biological applications. It is usually very difficult to interpret the output of a given model from a biological point of view, which limits the application of the model

Division of labor:

Maria ilyas	Data Gathering and Preprocessing
Muhammad Shah Zaib	Classification (Coding)
Hamna Imran	Model Evaluation ,Model Validation

We all write documentation of our individual tasks .