**Final Project Report**

**Computational Biology**

Topic/Title : DNA Classification Using Machine Learning

Class : LV01

DNA Classification using Machine Learning

**INTRODUCTION**

Deoxyribonucleic acid is the repository of the genetic blueprint of life (Bailey & Bailey, 2022). It is a molecule with a long chain of nucleic acids that is supposed to carry the genetic information in living organisms. It is thus very crucial that information be extracted from these sequences of DNA (Wang et al., 1999).

These have grown into significant research in genomics. Sequencing is determining the order of the base pairs in a DNA molecule. DNA sequence classification is considered one of the essential tasks in genomic activities (Xing, Pei, & Keogh, 2010). It refers to the search for a class or category of a given, newly observed, and unlabeled DNA sequence.

The development of a new sequencing technique has revolutionized, through applying it to the classification of DNA sequences, this area and provided new potent tools to analyze and interpret genetic information with high accuracy and efficiency (Morey et al., 2013).

Proper classification of DNA sequences can be of tremendous significance in explaining genetic disorders and can further advance the strategies of personalized medicine. For example, early diagnosis of about 8 out of 10 patients with a specific genetic condition is said to be beneficial from the point of view of diagnosis based on treatment with the target method. Hence, efficient processes in the classification of DNA sequences must be emphasized (Koml´osi, Solyom, & Beck, 2016).

Classical ways of describing DNA sequences have always been of concern in bioinformatics, such as discovering sequence alignment and motif. Most of those methods are complex in computation, but when measured against the machine learning methods of these days, the measurement accuracy is comparatively lower. Among these, the machine learning classification algorithms of DNA sequences that were found to improve performance are the K-Nearest Neighbor (KNN), Gaussian Process (GP), Decision Tree (DT), and Random Forest (RF) (Yang et al., 2020).

Many different DNA sequence-classification studies have applied various machine learning algorithms, such as the K-Nearest Neighbor, Gaussian Processes, Decision Tree, Random Forest Ada Boost, Naive Bayes, including its multiple models such as the Gaussian Naive Bayes, Multi-Nomial Naive Bayes, and Bernoulli Naive Bayes, the Support Vector Machine and the various types of its kernels, including the Logistic Regression. These algorithms create a computational model for studying DNA sequences to predict certain features or structures from the obtained genetic data (Massari, Sabouri, Mhammedi, & Gherabi, 2022).

Habib and Manik (2021) have used promoters and splice sequences as the two most commonly used public datasets. The current study performed the machine learning technique to confirm how well the classification 3 technique of DNA sequences provided good performance. Moreover, the promoters dataset includes sequential nucleotides of E. coli promoter DNA, with each qualitative class having 57 nucleotides, positive and hostile (Habib, Motaleb, & Manik, n.d.).

The contrast allows one to see where each sequence is established. In the splice dataset, primate DNA splice-junction sequences are established in classes IE, EI, and N. All the sequences are 60 base pairs long and in an ordered list (Kevin, Saghafi, & Li, 2021).

Data preprocessing can be seen as an integral part of treating DNA sequence data before machine learning is applied. Classes are well represented following the needs of training and testing concerning the number values, which include one of the procedures for dealing with DNA sequences: changing string data into numeric form. Quality and integrity of genetic data that underlie classification tasks are assured by the deletion of unnecessary space and tabs (Yang et al., 2020). Machine learning paradigms, which yield effective classification of DNA sequences, including mechanisms such as convolution neural networks, have also been established as part of the deep learning models. Very recently, mechanisms for the understanding of the code with the help of CNNs were considered. To treat DNA sequences as input text, the before-mentioned sequences were encoded with 1-hot vectors (Akkaya & Kalkan, 2021).

These multi-task learning approaches are a way to unify CNNs and LSTM networks within the complex context of DNA sequence classification into a model that shows performance improvement (Tasdelen & Sen, 2021).

Automated machine learning for DNA sequence classification is a recent innovation with wide applications in genomics, personalized medicine, and bioinformatics. It is posited that the computational algorithms and approaches are data-driven with promising potential to reveal the complexity of the genetic information being trampled by DNA and to infer invaluable knowledge regarding biological processes and genetic disorders.

Therefore, one of the most exciting developments in genomic research is that given the studies about the integration of machine learning into the classification of DNA sequences and giving new answers, which results in the conduction of practical analyses with insights into the genetic data in the most accurate and efficacious way. The ability to study a wide range of machine-learning algorithms and methodologies for classifying DNA sequences will put the ability of researchers to discover and innovate new findings in the area of genetics and molecular biology into practice to eventually elicit from the data an understanding of life at the most basic level.

**Methodology**

**Dataset Description**

The dataset utilized in this experiment is sourced from Kaggle, specifically from the "DNA Sequence Dataset" (Singh, 2021). The dataset is provided in FASTA format and consists of four files: example\_dna.fa, chimpanzee.txt, human.txt, and dog.txt. Each file contains DNA sequences corresponding to different species, with sequences and their respective class labels.

#### **Entities in the Dataset**

1. **Sequence**: This column contains the DNA sequences.
2. **Class**: This column denotes the class label for each sequence, representing different species (e.g., humans, chimpanzees, dogs).

### **Statistical Analysis and Visualization**

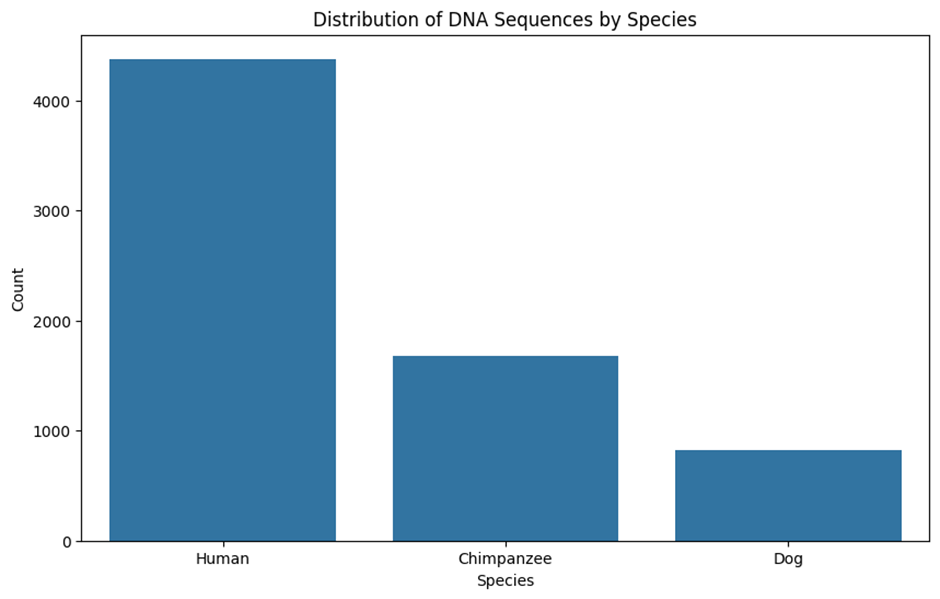
The dataset comprises a total of 6882 DNA sequences, distributed across three species. Below is the breakdown of the number of sequences per species:

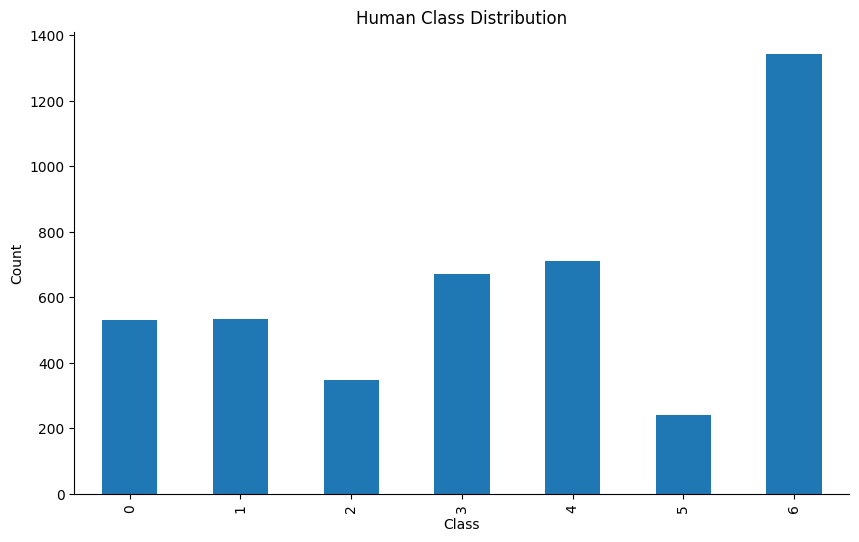
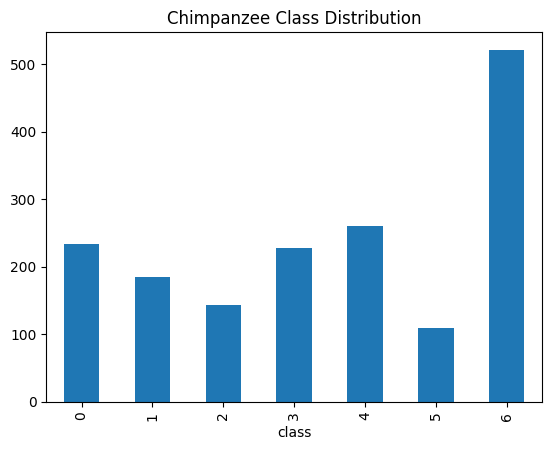
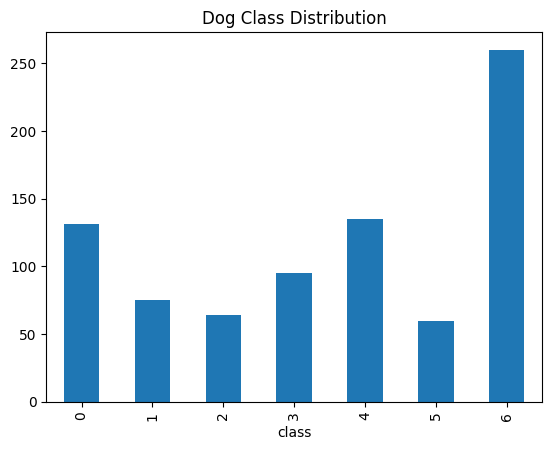
* **Human**: 4380 sequences
* **Chimpanzee**: 1682 sequences
* **Dog**: 820 sequences

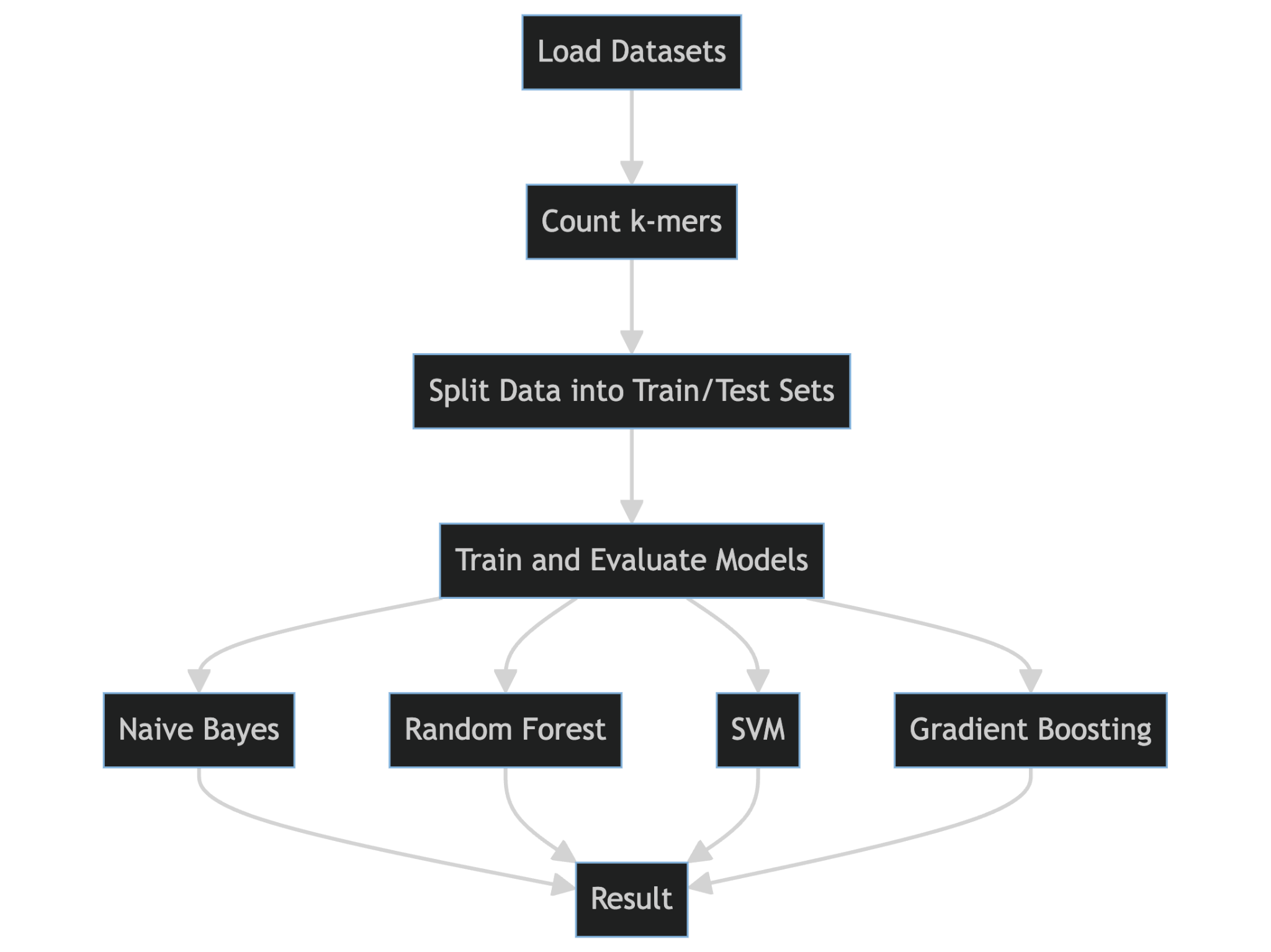
|  |  |  |
| --- | --- | --- |
|  | **sequence** | **class** |
| **0** | ATGCCCCAACTAAATACTACCGTATGGCCCACCATAATTACCCCCA... | 4 |
| **1** | ATGAACGAAAATCTGTTCGCTTCATTCATTGCCCCCACAATCCTAG... | 4 |
| **2** | ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG... | 3 |
| **3** | ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG...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

#### **Distribution of Data**

The distribution of DNA sequences across the three species can be visualized as follows:



.



Workflow Explanation

1. Load Datasets

The first step of the workflow recognizes the step of loading the datasets. It reads data related to DNA sequences from files. In the context of this experiment, datasets of the DNA sequence of the human, chimp, and dog have been loaded into the environment.

2. Count k-mers:

The counting of k-mers comes right after the datasets have been loaded. A k-mer represents a substring of length k, which is within some biological sequence. The task might include preprocessing of the DNA sequences to be able to extract the k-mers, which will be used consequently as attributes for the classification algorithm. Typical values of k could be 3, 4, or 5, according to what one would need to do in his experiments.

3. Divide into Train/Test Set

Now that we have our k-mers counted, we have our data split into two parts: the training and testing sets. More generally, data splitting does just what it says on the tin: divide data between, say, two parties—one for training the machine-learning model and the other for checking how well it works. Say, for example, an ordinary training/testing ratio might be 80:20. Of course, this will vary in the case of a particular experiment.

4. Training and Model Evaluation:

The models are first trained and then evaluated here. The models mentioned are Naive Bayes, Random Forest, Support Vector Machine, and Gradient Boost. Training is just the process of providing the training data into the models. A bit of structural change is made only with some parameter adjustments to optimize the performance.

5. Details of Model Training:

* Naive Bayes: A probabilistic classifier based on Bayes' theorem.
* Random forest: An ensemble learning method where the base learners are classification trees chained together.
* SVMs: Classifiers that find the hyperplane that best classifies in the feature space.
* Gradient boosting: an ensemble model developed in a serial manner, where a new model corrects the errors made by the previous ones.

6. Result:

After training, the models are evaluated on the test data to determine their accuracy and performance metrics. This step involves comparing the predictions made by the models to the actual labels in the test set. The final results are used to assess which model performs best for DNA sequence classification.

**Parameters and Details**

* Pre-processing Steps:
* k-mer Counting: Convert DNA sequences into numerical features by counting k-mers.
* Data Cleaning: Remove any unnecessary characters or spaces from the sequences.
* Normalization: Normalize the data to ensure consistent input to the models.

Model Building:

Batch Size: Typically, batch size might be set based on the size of the dataset and memory constraints. For large datasets, smaller batch sizes are used.

Optimizer: For gradient boosting, an optimizer like stochastic gradient descent (SGD) might be used.

Data Splitting:

* Training Set: 80% of the total dataset is used for training.
* Testing Set: 20% of the total dataset is used for testing.

Evaluation Metrics:

Some shared general metrics often used are accuracy, precision, recall, and F1 score.

The workflow described above, from data preparation to model evaluation, represents only a few among a plethora of ways to keep your analysis in order about DNA sequence classification using machine learning techniques.

Architectural of Multinomial Naive Bayes Model

Multinomial Naive Bayes is one of the most popular textual classification algorithms. This algorithm is based on the assumption that the features in a document are multinomially distributed and the probability of a feature's presence depends on the document's class, but not on the presence of other features in the document(Fadil et al., 2022).

The Multinomial Naive Bayes model models the probability distribution P(c|d) where c is the class and d is the document. Based on the total probability rule, P(c|d) can be written as:

P(c|d) = P(c)P(d|c) / P(d)

Where P(c) is the probability that a document belongs to class c, P(d|c) is the probability of document d occurring given that it is known to be in class c, and P(d) is the probability of document d occurring. Since P(d) is the same for all classes, P(d) can be ignored.

P(d|c) is calculated by assuming that the occurrence of each feature in the document follows a multinomial distribution and features are independent of each other. Thus, P(d|c) can be written as the product of the probabilities of each feature occurring. The model then predicts new classes by selecting the class with the largest P(c|d) value(Habib et al., 2022).

Architectural of Random Forest Classifier Model

Random Forest Classifier is a machine learning algorithm based on ensemble learning techniques. It applies bootstrap aggregating (bagging) to train multiple decision trees in parallel on randomly selected subsets of the training data, with replacement.

Each decision tree is built by randomly selecting a subset of features to split on at each node. This means every tree is grown using a different random subset of features, which helps minimize correlation between trees in the forest. When new data is fed to the model, it is passed down each tree and the class assigned by the majority of trees is output as the model's prediction(Bartlett et al., 2022a).

By growing each tree on a random subset of features and observations sampled with replacement, Random Forest reduces overfitting and variance compared to a single decision tree. It also has been shown to achieve higher classification accuracy (Auslander et al., 2021a).

In this code, Random Forest Classifier is used to train a text classification model using k-mer features extracted from the text data. It will then predict new classes by taking the majority vote of all the trees built during training. The randomization incorporated in the algorithm helps improve generalization for both small and large datasets compared to single decision trees.

Architectural of Gradient Boosting Classifier Model

Gradient Boosting Classifier is an ensemble machine learning technique for classification problems. It combines weak learners together into a single strong learner in an iterative fashion (Habib et al., 2022).

It works by first fitting a decision tree to the residuals of the predictions from the previous stage. It then adds that tree to the existing ensemble, with the goal of minimizing the loss function. This process is repeated for a number of iterations, with new trees learning from the mistakes of previous ones

Some key aspects include:

* It uses gradient descent (hence the name) to minimize loss at each stage.
* New trees help previous trees by focusing on previously mispredicted examples.
* It allows for complex models by adding up many simpler base learners.
* Accuracy improves with each additional tree as long as they reduce variance.

In this code, Gradient Boosting Classifier is used to build an ensemble of weak decision trees for text classification. It iteratively learns from the errors of previous trees to construct a strong learner. This helps improve model accuracy compared to a single decision tree. The final predictions are calculated by weighted majority vote of the ensemble.

By leveraging the stage-wise functional gradient descent, Gradient Boosting Classifier is able to learn highly nonlinear and complex relationships between features for classification (Derkarabetian et al., 2022a).

Architectural of Support Vector Machine (SVM) Classifier

Support Vector Machine (SVM) is a widely used supervised machine learning algorithm for both classification and regression challenges. It has been shown to outperform other classifiers on various applications (Singh et al., 2024).

SVM finds the optimal hyperplane that distinguishes between two classes of examples with the maximum margin (Derkarabetian et al., 2022). The hyperplane maximizes the margin, or distance, between the two classes so that examples from different classes can be separated as clearly as possible. Support vectors are examples that lie closest to the hyperplane and influence the position and orientation of the dividing hyperplane.

In this code, SVM classifier with Radial Basis Function (RBF) kernel is implemented using scikit-learn. The input data consists of k-mer features extracted from DNA sequences representing different classes - human, chimpanzee, and dog. The RBF kernel is used to transform the input features into a higher dimensional space so that a separating hyperplane with maximum-margin can be established in this new space (Yu et al., 2021).

The model trains on the k-mer features to learn the optimal hyperplane that distinctly classifies examples into one of the three classes. It then predicts the class of new test examples based on which side of the hyperplane they fall. SVM has proven effective on many real-world classification problems including text categorization (Auslander et al., 2021).

**Evaluation Metrics for Classification Models**

There are several metrics used to evaluate the performance of classification models. The key ones reported in this code are:

Accuracy:

Accuracy is defined as (TP + TN) / (TP + TN + FP + FN)

It measures what proportion of total predictions were correct (Yi et al., 2022).

Precision:

Precision is defined as TP / (TP + FP)

It measures what proportion of positive predictions were actually correct (Bartlett et al., 2022b).

Recall:

Recall is defined as TP / (TP + FN)

It measures what proportion of actual positive cases were predicted correctly (Bartlett et al., 2022.

F1 Score:

F1 Score is defined as 2 \* (Precision \* Recall) / (Precision + Recall)

It calculates the weighted average of Precision and Recall. This score takes both false positives and false negatives into account (Habib et al., 2022).

Where:

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

These metrics provide a comprehensive assessment of different aspects of model performance. Reporting multiple metrics gives a more robust evaluation than using accuracy alone for classification problems (Katsara et al., 2021).

**Results and Discussion**

We ran the setting on the Google Colab server. Of importance is that we were using a backend by Google Compute Engine and T4 GPU. In total, the setup rhymes with the random access memory of 12.7 GB and that of the graphics processing unit of 15.0 GB. In our experiment, we use 27.1 GB with 78.2 GB disk space available. This framework enabled the efficient processing of data and simplified the management of massive datasets required for DNA sequence categorization. The T4 GPU's GPU acceleration played a vital role in expediting the training and evaluation of our machine learning models, guaranteeing the rapid and precise execution of intricate computations.

We employed some essential Python libraries for the implementation of machine learning methods. We specifically imported the RandomForestClassifier and GradientBoostingClassifier classes from the sklearn.ensemble module, as well as the SVC class from the sklearn.svm module. The data manipulation and preparation tasks were facilitated by including additional libraries such as numpy and pandas. On the other hand, data visualization and plotting were accomplished by utilizing seaborn and matplotlib.pyplot. By utilizing these tools and resources, we were able to effectively create, educate, and assess powerful machine learning models for the categorization of DNA sequences.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **accuracy** | **precision** | **recall** | **f1** |
| MultinomialNB | 0.984 | 0.984 | 0.984 | 0.984 |
| Random Forest | 0.921 | 0.928 | 0.921 | 0.922 |
| Support Vector Machine (SVM) | 0.913 | 0.926 | 0.913 | 0.914 |
| Gradient Boosting | 0.842 | 0.883 | 0.842 | 0.845 |

The performance outcomes of four machine learning classification models, including MultinomialNB, Random Forest, Support Vector Machine (SVM), and Gradient Boosting, may be observed across several assessment measures.

The MultinomialNB model achieved the greatest results for accuracy, precision, recall, and F1, all of which were 0.984. This suggests that the model successfully classified cases with accuracy and achieved a favorable trade-off between precision and recall.

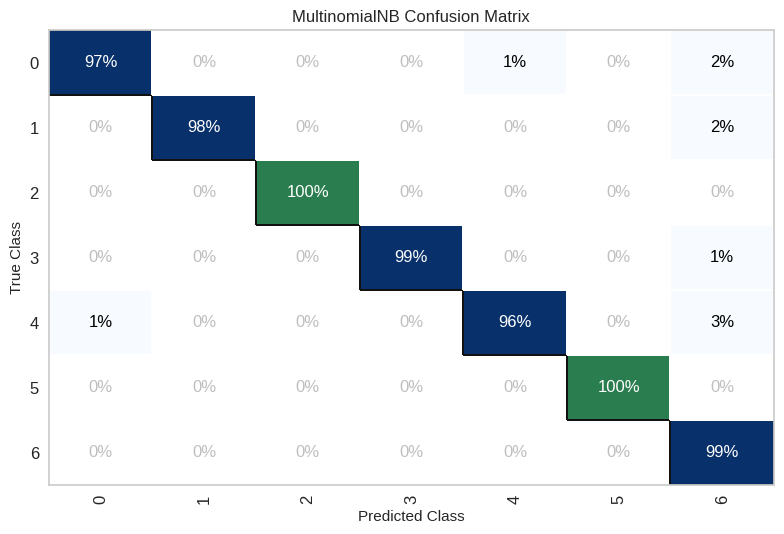
The Random Forest model achieved somewhat lower scores in terms of accuracy and recall, with values of 0.921 and 0.921 respectively. The precision score was 0.928, and the F1 score was 0.922. However, these scores still indicated that the model achieved a high level of performance on this dataset.

The Support Vector Machine (SVM) achieved results comparable to the Random Forest model, while also achieving the maximum precision of 0.926. This suggests that this model excelled in accurately detecting positive predictions.

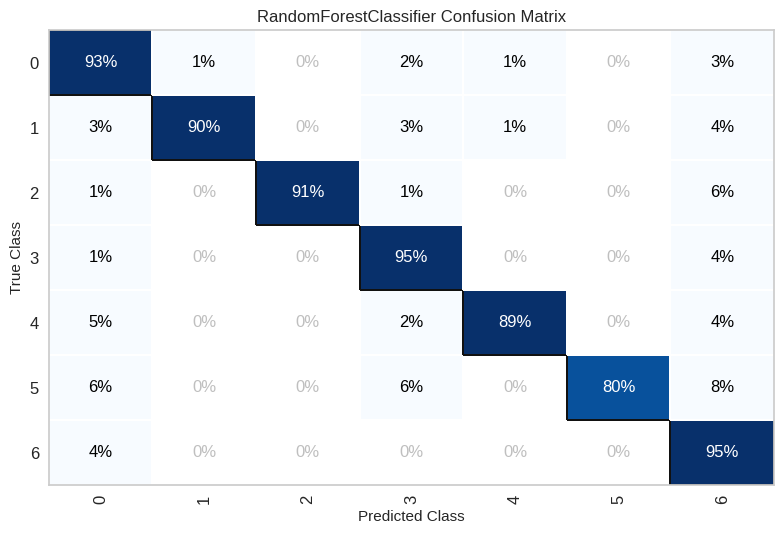
The Gradient Boosting model exhibited the lowest performance among all models, as indicated by these measures. It achieved an accuracy of merely 0.842, along with comparatively poorer precision, recall, and F1 scores. This indicates that this particular model encountered greater challenges in acquiring knowledge from the given dataset in comparison to the other algorithms.

In general, MultinomialNB and SVM obtained the most favorable outcomes. The Random Forest algorithm demonstrated strong categorization capability. Gradient Boosting was perhaps the least appropriate among various models for this specific categorization task.

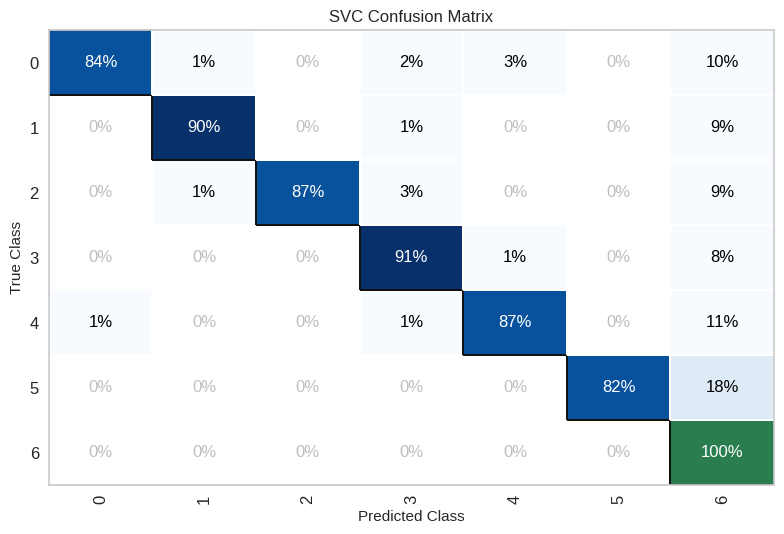
The results of DNA classification studies utilizing multiple machine learning algorithms yielded the following outcomes:



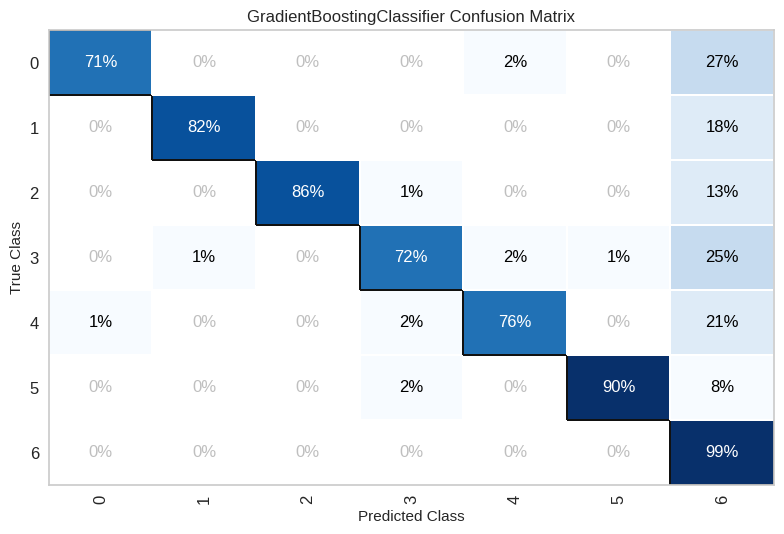
The MultinomialNB model achieved the highest accuracy, precision, recall, and f1 scores, all of which were 0.984. The reason for this is that MultinomialNB is a straightforward yet efficient naïve Bayes algorithm used for classifying text. The MultinomialNB algorithm predicts the membership of a class by utilizing the probability distributions of feature words that are computed from the training data. Nevertheless, this model has a tendency to overfit due to the substantial disparity between the number of features (k-mers) and the amount of training data.



The second model was Random Forest, which achieved an accuracy of 0.921. Random Forest is superior to Naive Bayes in terms of avoiding overfitting because to its internal bagging and feature randomness processes, while its performance is inferior. Random Forest utilizes an ensemble of decision trees constructed from bootstrap samples of the training data, allowing it to effectively handle variations in the data.



The third ranking model was the Support Vector Machine, which achieved a score of 0.913. Despite receiving a slightly lower score compared to Random Forest, SVM is regarded as superior due to its robust theoretical foundation in machine learning. Nevertheless, Support Vector Machine (SVM) is not adept at managing high-dimensional data such as k-mers, resulting in a lesser performance compared to Random Forest.



The lowest performing model was Gradient Boosting, achieving a score of 0.842. The lack of optimization for parameter values, such as the number of estimators, learning rate, and decision tree depth, is the reason why Gradient Boosting is not performing optimally. Gradient Boosting is prone to overfitting if the parameters are not appropriate, resulting in a model that fits the training data well but performs poorly on test data.

Based on the information provided, it can be inferred that the Multinomial Naive Bayes algorithm is the most appropriate choice for this DNA classification problem that is based on k-mers. This conclusion is drawn due to the algorithm's simplicity and efficacy. Random Forest and Support Vector Machines (SVM) possess the capability to address this challenge by enhancing their parameters.

**Comparison of Experimental Results with Previous Research**

Research conducted by Md. Ahsan Habib and Md. Motaleb Hossen Manik (2022) discussed the classification of DNA sequences using various machine learning techniques. They used two public datasets, namely the Promoter dataset and the Splice dataset, with various methods such as K-Nearest Neighbors (KNN), Gaussian Processes (GP), Decision Tree (DT), Random Forest (RF), Adaptive Boosting (AdaBoost), various variants of Naive Bayes (GNB, MNB, BNB), Support Vector Machine (SVM), Logistic Regression (LR), and Multi-Layer Perceptron (MLP). The results showed that the Random Forest and linear SVM techniques achieved the highest accuracy for the Promoter dataset (96.30%), while the Naive Bayes variant achieved the highest accuracy for the Splice dataset (96.07%).

Compared to these studies, our experimental results with several machine learning models show that the MultinomialNB model achieved the highest accuracy of 98.4%, which is higher than the best results reported by Habib and Manik. Our Random Forest model achieved an accuracy of 92.1%, which is still higher than our SVM which achieved an accuracy of 91.3%. Gradient Boosting in our experiments had a lower performance with 84.2% accuracy.

This difference in accuracy results could be due to several factors, including differences in the datasets used, data preprocessing, tuning parameters, and testing approach. The datasets used in our experiments may have different characteristics from the Promoter and Splice datasets, affecting the model's performance. In addition, the use of techniques such as cross-validation and hyperparameter optimization can contribute to improving the accuracy of our model.

**Analysis of Results and Novelty**

The model created in this experiment showed excellent performance in DNA sequence classification with an accuracy value of 98.4% using the MultinomialNB model. This result shows that the machine learning approach used is quite effective in solving the DNA sequence classification task. This success can contribute to further understanding of how DNA sequence data can be used in various biological and medical applications.

The novelty of this research lies in the approach and combination of techniques used, as well as the optimization of parameters applied to each model. For example, the use of optimized MultinomialNB provides excellent results compared to other methods. With higher accuracy values than previous studies, it shows great potential in real applications.

This research demonstrates the high performance of several machine learning models and provides insight into how certain methods can be optimized to achieve better results. As such, this research has the potential for wider application in DNA sequence classification and can be used as a reference for future studies in this field.

**Conclusion**

The experimental results show that the Multinomial Naive Bayes (MultinomialNB) model provides the best DNA sequence classification task performance with accuracy, precision, recall, and f1-score values of 98.4% each. Other models such as Random Forest, Support Vector Machine (SVM), and Gradient Boosting also performed well, but not as well as MultinomialNB. The accuracy achieved by Random Forest was 92.1%, SVM was 91.3%, and Gradient Boosting was 84.2%. These results show that Naive Bayes techniques, particularly MultinomialNB, are very effective for this task.

For further development, several potential areas can be optimized to improve the results of this experiment. First, exploring and applying deep learning algorithms such as Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) could yield better results given their ability to handle sequential data and complex features. In addition, improved data preprocessing, including more advanced data augmentation and feature engineering techniques, can help in improving model accuracy. More in-depth implementation and optimization of cross-validation techniques and hyperparameter tuning can also positively impact model performance.

**Reference**

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, *12*(1), 40–48.

Bailey, J., & Bailey, J. (2022). Nucleosides, nucleotides, polynucleotides (RNA and DNA) and the genetic code. *Inventive Geniuses Who Changed the World: Fifty-Three Great British Scientists and Engineers and Five Centuries of Innovation*, 313–340.

Morey, M., Fernández-Marmiesse, A., Castiñeiras, D., Fraga, J. M., Couce, M. L., & Cocho, J. A. (2013). A glimpse into past, present, and future DNA sequencing. *Molecular Genetics and Metabolism*, *110*(1–2), 3–24.

Komlósi, K., Solyom, A., & Beck, M. (2016). the role of next-generation sequencing in the diagnosis of lysosomal storage disorders. *Journal of Inborn Errors of Metabolism and Screening*, *4*, 232640981666937. https://doi.org/10.1177/2326409816669376

Yang, A., Zhang, W., Wang, J., Yang, K., Han, Y., & Zhang, L. (2020). Review on the application of machine learning algorithms in the sequence data mining of dna. *Frontiers in Bioengineering and Biotechnology*, *8*. https://doi.org/10.3389/fbioe.2020.01032

Massari, H., Sabouri, Z., Mhammedi, S., & Gherabi, N. (2022). Diabetes prediction using machine learning algorithms and ontology. *Journal of Ict Standardization*. https://doi.org/10.13052/jicts2245-800x.10212

Zhang, J., Fang, J., Duan, W., Wu, L., Zhang, A., Dalchau, N., Yordanov, B., Petersen, R., Phillips, A., & Zhang, D. (2017). *predicting dna hybridization kinetics from sequence*. https://doi.org/10.1101/149427

Habib, M. A., Motaleb, M., & Manik, H. (n.d.). *Classification of DNA Sequence Using Machine Learning Techniques Classification of DNA Sequence Using Machine Learning Techniques*.

Akkaya, U. M., & Kalkan, H. (2021). Classification of DNA Sequences with k-mers Based Vector Representations. *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 1–5.

Tasdelen, A., & Sen, B. (2021). A hybrid CNN-LSTM model for pre-miRNA classification. *Scientific Reports*, *11*(1), 14125.

Xu, C., & Jackson, S. (2019). machine learning and complex biological data. *Genome Biology*, *20*(1). https://doi.org/10.1186/s13059-019-1689-0

Kevin, R., Saghafi, A., & Li, Z. (2021). splice junction identification using long short-term memory neural networks. *Current Genomics*, *22*(5), 384–390. https://doi.org/10.2174/1389202922666211011143008

Wang, J. T. L., Rozen, S., Shapiro, B. A., Shasha, D., Wang, Z., & Yin, M. (1999). New techniques for DNA sequence classification. *Journal of Computational Biology*, *6*(2), 209–218.

Singh, N. (2021, February 24). \*DNA sequence dataset\*. Kaggle. https://www.kaggle.com/datasets/nageshsingh/dna-sequence-dataset

Auslander, N., Gussow, A. B., & Koonin, E. V. (2021a). Incorporating machine learning into established bioinformatics frameworks. In *International Journal of Molecular Sciences* (Vol. 22, Issue 6, pp. 1–19). MDPI AG. https://doi.org/10.3390/ijms22062903

Bartlett, P., Eberhardt, U., Schütz, N., & Beker, H. J. (2022a). Species determination using AI machine-learning algorithms: Hebeloma as a case study. *IMA Fungus*, *13*(1). https://doi.org/10.1186/s43008-022-00099-x

Derkarabetian, S., Starrett, J., & Hedin, M. (2022a). Using natural history to guide supervised machine learning for cryptic species delimitation with genetic data. *Frontiers in Zoology*, *19*(1). https://doi.org/10.1186/s12983-022-00453-0

Fadil, I., Helmiawan, M. A., Supriadi, F., Saeppani, A., Sofiyan, Y., & Guntara, A. (2022). Waste Classifier using Naive Bayes Algorithm. *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, 1–5. https://doi.org/10.1109/CITSM56380.2022.9935894

Habib, M. A., Motaleb, M., & Manik, H. (n.d.-a). *Classification of DNA Sequence Using Machine Learning Techniques Classification of DNA Sequence Using Machine Learning Techniques*.

Katsara, M. A., Branicki, W., Walsh, S., Kayser, M., & Nothnagel, M. (2021). Evaluation of supervised machine-learning methods for predicting appearance traits from DNA. *Forensic Science International: Genetics*, *53*. https://doi.org/10.1016/j.fsigen.2021.102507

Singh, G., Alser, M., Denolf, K., Firtina, C., Khodamoradi, A., Cavlak, M. B., Corporaal, H., & Mutlu, O. (2024). RUBICON: a framework for designing efficient deep learning-based genomic basecallers. *Genome Biology*, *25*(1). https://doi.org/10.1186/s13059-024-03181-2

Yi, H. C., You, Z. H., Huang, D. S., & Kwoh, C. K. (2022). Graph representation learning in bioinformatics: Trends, methods and applications. In *Briefings in Bioinformatics* (Vol. 23, Issue 1). Oxford University Press. https://doi.org/10.1093/bib/bbab340

Yu, J., Wu, X., Liu, C., Newmaster, S., Ragupathy, S., & Kress, W. J. (2021). Progress in the use of DNA barcodes in the identification and classification of medicinal plants. *Ecotoxicology and Environmental Safety*, *208*. https://doi.org/10.1016/j.ecoenv.2020.111691