Codon Usage Predict Taxonomic Identity and Genetic Composition

Author: Qingcheng Yu (qy2281), Zhihao Yi (zy2523), Haotian Yang(hy2755), Hollis Chen(yc4140)

Introduction

In this study we investigate how to find out whether we can classify some properties of nucleic acids from the usage of different synonymous codons rather lager calculations from each nucleotide sequence themself. The genome DNA coding describes 64 different codons of the proteins of the organism, which map to 21 different amino acids and a stop signal, and different organisms have different amino acid sequences of their proteins. They use the synonymous condos for different amino acids. So those coding DNA sequences can carry lots of information that exceeds the need for encoding amino acid sequences. We conduct analysis of existing genetic datasets, and we use machine learning, and neural networks to train over 13,000 organisms' data. Our modules show that it is possible to identify an organism's DNA type(mitochondrial, nuclear, chloroplast) and taxonomic identity simply by using 64 codon usage frequencies of its own genetic code.

Data Set and Paper

In this paper, we use the Dataset in UCI Machine learning Repository(Codon usage Data Set): https://archive.ics.uci.edu/ml/datasets/Codon+usage [1]. This data set examined Codon usage frequencies in the genomic coding DNA of a large sample of diverse organisms from different taxa tabulated in the GUTG database. There are 514 different genus names, and the classification by genus includes bacteria, invertebrates, mammals, plants, primates, rodents, and vertebrates.

There are 69 columns:

- Column 1: Kingdom (3-letter)
- Column 2: DNAtype (Integer)
- Column 3: SpeciesID (Integer, which uniquely indicates the entries of an organism).
- Column 4: Ncodons (Algebraic sum of the numbers listed for the different codons in an entry of CUTG)
- Column 5: SpeciesName (Descriptive string for each species)
- Columns 6-69: Codon (5 digits floating point number)

The paper uses Naive Bayes, K-nearest neighbor, Random Forest, Artificial Neural Network, and XGBoost to find out which method is the best in this hypothesis. First, there is some preprocessing for the dataset. They delete the missing value and inconsistent value rows and convert the kingdom column from string type to int type. Next, they remove any rows that do not belong to these five classes to classify the five kingdom classes of the organisms (archaea, bacteria, eukaryote, bacteriophage, and virus). They also remove any rows that do not belong to these three classes, so we find the three DNA types(genomic, mitochondrial, and Chloroplast). In conclusion, this paper claims all 5 methods in DNA type work, and KNN has better precision,

XGBoost, and Random Forest have better model fit [2].

Models and Results Reproduction

K-Nearest Neighbors

KNN is a machine learning algorithm that is used for classification and regression tasks by predicting the value of a target variable based on the values of its nearest neighbors in n-dimensional feature space [3]. It uses the Euclidean distance metric to calculate the differences between data observations and group similar objects together. The algorithm is non-parametric and does not make assumptions about the underlying data distribution. While KNN is simple to implement and understand, it can be sensitive to the scale of the data and may not perform well on high-dimensional datasets. And the formula is shown below:

$$dist_{Euc}(p,q) = \sqrt{\sum_{j=1}^{n}(p_j - q_j)^2}$$

To find the best hyperparameter for a KNN model using 5-fold cross-validation, we followed a similar process to find the best hyperparameter for any machine learning model using this model. First, we split the data into five folds and used four folds for training the model and one field for validation. We then repeated this process for each fold using a different fold for validation each time. After completing all five folds, we averaged the results to determine the model's overall performance.

One of the hyperparameters that we considered tuning for a KNN model is the number of nearest neighbors, which is typically referred to as ‘K’. By using 5-fold cross-validation, we got a more reliable estimate of the model's performance and identified the best combination of hyperparameters for our dataset and mode. This can help us to improve the model’s accuracy and predictive power.

For the Kingdom label, we got the best hyperparameters {‘leaf_size’: 1, ‘n_neighbors’: 4}, and the overall scores and confusion matrix are shown below in Figure 1, respectively. For DNA type labels, we got the best hyperparameters {‘leaf_size’: 1, ‘n_neighbors’: 4} as well, and the overall scores, confusion matrix, and classification report are shown below in Figure 2, respectively.

```
accuracy_score:  0.9228692634014655
precision_score:  0.9225455639772154
recall_score:  0.9228692634014655
f1_micro:  0.9228692634014655
f1_macro:  0.8063484795181969
auc_score:  0.9468160283721035
```

```
Confusion matrix :
 [[  11    8    0    0    2]
 [   4  499   18   10   13]
 [   0   28 1343    3   41]
 [   0   12    2   29    0]
 [   1   12   46    0  511]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.69      0.52      0.59        21
           1       0.89      0.92      0.90       544
           2       0.95      0.95      0.95      1415
           3       0.69      0.67      0.68        43
           4       0.90      0.90      0.90       570

    accuracy                           0.92      2593
   macro avg       0.83      0.79      0.81      2593
weighted avg       0.92      0.92      0.92      2593
```

Figure 1. The classification report for the best k-NN model for Kingdom.

```
accuracy_score:  0.9919012726571539
precision_score:  0.9919016117143766
recall_score:  0.9919012726571539
f1_micro:  0.9919012726571539
f1_macro:  0.9788090107927635
auc_score:  0.9905754241590893

Confusion matrix :
 [[1848    0    4]
 [   5  570    4]
 [   5    3  154]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1852
           1       0.99      0.98      0.99       579
           2       0.95      0.95      0.95       162

    accuracy                           0.99      2593
   macro avg       0.98      0.98      0.98      2593
weighted avg       0.99      0.99      0.99      2593
```

Figure 2. The classification report for the best k-NN model for DNA type.

In summary, the results from the original paper show higher accuracy, AUC, and macro-F1 scores for both kingdom and DNA-type classification tasks compared to the results obtained in the current study. The accuracy for kingdom classification in the original paper was 0.9660, while the current study obtained a slightly lower result of 0.9229. The AUC score for kingdom classification in the original paper was 0.9792, while the current study obtained a slightly lower result of 0.9468. The macro-F1 score for kingdom classification in the original paper was 0.9827, while the current study obtained a slightly lower result of 0.9229. For DNA-type classification, the results from the original paper also showed higher accuracy, AUC, and macro-F1 scores compared to the current study. The accuracy in the original paper was 0.9942, while the current study obtained a slightly lower result of 0.9919. The AUC score in the original paper was 0.9997, while the current study obtained a slightly lower result of 0.9907. The macro-F1 score in the original paper was 0.9867, while the current study obtained a slightly lower result of 0.9788.

Random Forests

Random Forest (RF) is an ML algorithm that combines the predictions of multiple decision trees by voting to make a final prediction. It is an ensemble method that can help to improve the accuracy of the model by reducing variance and bias introduced by individual decision trees [4]. RF classifiers are useful for avoiding overfitting and working with large, high-dimensional datasets. They make predictions based on a random selection of predictors and training data. Overall, RF is a powerful tool for classification and regression tasks.

One of the hyperparameters that we considered tuning for an RF model is the number of max depth in the tree. By using 5-fold cross-validation, we got a more reliable estimate of the model's performance and identified the best combination of hyperparameters for our dataset and mode. This can help us to improve the model's accuracy and predictive power.

For the Kingdom label, we got the best hyperparameters {'max_depth': 14} as well, and the overall scores, confusion matrix, and classification report are shown below, respectively:

```
accuracy_score:  0.9028152718858465
precision_score:  0.902779527615017
recall_score:  0.9028152718858465
f1_micro:  0.9028152718858465
f1_macro:  0.7048699252444519
auc_score:  0.9785524068756736
```

```
Confusion matrix :
 [[   4   10    5    0    2]
 [   4  508   24    1    7]
 [   0   45 1308    0   62]
 [   0   20    6   17    0]
 [   1   18   47    0  504]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.44      0.19      0.27        21
           1       0.85      0.93      0.89       544
           2       0.94      0.92      0.93      1415
           3       0.94      0.40      0.56        43
           4       0.88      0.88      0.88       570

    accuracy                           0.90      2593
   macro avg       0.81      0.67      0.70      2593
weighted avg       0.90      0.90      0.90      2593
```

Figure 3. The classification report for the best RF model for Kingdom.

For DNA type labels, we got the best hyperparameters {'max_depth': 10} as well, and the overall scores, confusion matrix, and classification report are shown below, respectively:

```
accuracy_score:  0.9845738526802931
precision_score:  0.9846095035922233
recall_score:  0.9845738526802931
f1_micro:  0.9845738526802931
f1_macro:  0.9654932824647514
auc_score:  0.9986862436565255
```

```
Confusion matrix :
 [[1850    1    1]
 [  16  561    2]
 [  17    3  142]]
```

Classification report :

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1852
1	0.99	0.97	0.98	579
2	0.98	0.88	0.93	162
accuracy			0.98	2593
macro avg	0.98	0.95	0.97	2593
weighted avg	0.98	0.98	0.98	2593

Figure 4. The classification report for the best RF model for DNA type.

In summary, the results from the original paper show higher accuracy, AUC, and macro-F1 scores for both kingdom and DNA-type classification tasks compared to the results obtained in the current study. The accuracy for kingdom classification in the original paper was 0.9298, while the current study obtained a slightly lower result of 0.9028. The AUC score for kingdom classification in the original paper was 0.9954, while the current study obtained a slightly lower result of 0.9786. The macro-F1 score for kingdom classification in the original paper was 0.8611, while the current study obtained a slightly lower result of 0.7049. For DNA-type classification, the results from the original paper also showed higher accuracy, AUC, and macro-F1 scores compared to the current study. The accuracy in the original paper was 0.9915, while the current study obtained a slightly lower result of 0.9846. The AUC score in the original paper was 0.9993, while the current study obtained a slightly lower result of 0.9987. The macro-F1 score in the original paper was 0.9832, while the current study obtained a slightly lower result of 0.9655.

Extreme Gradient Boosting

Extreme Gradient Boost (XGBoost) is a classification and method used in the original paper. As its name indicates, it is based on the Gradient Boost method, and it optimizes the forward stepwise manner by giving a learning rate η [5]. To achieve classification and prediction, The XGBoost is a tree method random forest (RF) method we used previously. However, it creates a lot of weak learner trees, which are way shallower than the random forest, and this method tweaks the trees step by step until it finds the best fit model for the dataset. Similar to gradient boost methods.

In the calculation, we will have a loss function (also called cost function) L(yi , γ), to minimize it, we will apply a stochastic gradient descent method to adjust the step model is using and finally returns a weighted average of our model of each step F(xi) [5].

In the code practice, the computer would use the data set input to generate an initial tree through this function:

$$F_0(x_i) = \underset{\gamma}{\operatorname{argmax}} \sum_{i=1}^{n} L(y_i, \gamma) = \underset{x_i}{\operatorname{argmax}} \sum_{i=1}^{n} (y_i - F(x_i))^2$$

After this tree is generated, the program would compute its residual r1 by using a function like this for future comparison in analyzing:

$$r_i = -\frac{\partial(\sum_{i=1}^{n}(y_i - f(x_i))^2)}{\partial F(x_i)} = -\frac{\partial(L(y_i, \gamma)}{\partial F(x_i)}$$

After this, the multiplier γ is calculated through this method below:

$$\gamma_m = \underset{\gamma}{\operatorname{argmax}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma_{m-1} \times h_m(x_i))$$

With the multiplier, the program can compute the next several trees one by one until it walks through all possible optimal trees. After all the possible trees have been looped through, the program would return a final result about the best tree. A flow chart can be found below

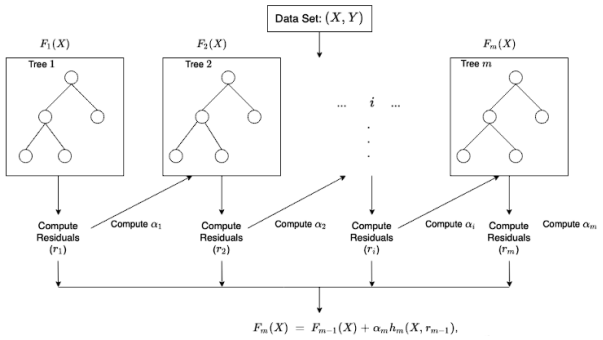


Figure 5. XGBoost flowchart [5].

We followed the process of the original paper, and put our processed dataset through the Python XGBoost package. As we did in previous packages, we also had five-fold cross-validations here, the max depth is 19.

```
accuracy_score:   0.907057462398766
precision_score:  0.905981332221538
recall_score:   0.907057462398766
f1_micro:   0.907057462398766
f1_macro:   0.7715236579360653
auc_score:   0.9743780760311666
```

```
Confusion matrix :
[[  18   12    1    1    3]
[   2  578   10    3    8]
[   0    1 1338    0    0]
[   0   21    0   20    6]
[   0    6    0    2  563]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.90      0.51      0.65        35
           1       0.94      0.96      0.95       601
           2       0.99      1.00      1.00      1339
           3       0.77      0.43      0.55        47
           4       0.97      0.99      0.98       571

    accuracy                           0.97      2593
   macro avg       0.91      0.78      0.82      2593
weighted avg       0.97      0.97      0.97      2593
```

Figure 6. The classification report for the best XGBoost model for Kingdom.

```
accuracy_score:   0.9873495834618945
precision_score:   0.9872954849749025
recall_score:   0.9873495834618945
f1_micro:   0.9873495834618945
f1_macro:   0.972479182248144
auc_score:   0.9979552185061434

Confusion matrix :
 [[2308    1    3]
 [   9  709    7]
 [   7    2  195]]

    Classification report :
                  precision    recall  f1-score   support

               0       0.99      1.00      1.00      2312
               1       1.00      0.98      0.99       725
               2       0.95      0.96      0.95       204

        accuracy                           0.99      3241
       macro avg       0.98      0.98      0.98      3241
    weighted avg       0.99      0.99      0.99      3241
```

Figure 6. The classification report for the best XGBoost model for DNA type.

As we can summarize from the result above, For kingdom classification results, the best model in the original paper demonstrates a high accuracy of 0.9502. At the same time, we got around 0.905, a bit lower than the paper. The paper got a high AUC of 0.997, and we got 0.974 for our AUC score, which is also a bit lower. Moreover, the paper got a macro-F1 score of 0.8846, and we have 0.771, similar, a bit lower. Similarly, on the DNA-type classification results, the paper yielded 0.9938. We have 0.9834 in the overall model accuracy, slightly lower. For the AUC value, the paper has 0.9997, while we have 0.9979, slightly lower. Finally, the paper has 0.986, and we have 0.972 in the macro-F1 score, which is also slightly lower.
Many possibilities that could cause the result of the differences shown above. Consider the reason, they may be from these aspects: different computing power differences causing the optimal tree-generated difference, the same equation used. Still, different realization methods in different coding languages and version differences: newer XGBoost uses a faster algorithm but will only develop 20 trees to find the optimum instead of 30~50 trees in previous versions. Finally, we may have different ways of processing our data, leading to differences.
According to the data we generated above and the confusion matrices above, we can still conclude that the XGBoost, as a popular method in the current market, is working well on getting better fit models for both DNA type and Kingdom data.

Artificial Neural Network

Artificial Neural Network (ANN) is a machine learning algorithm that consists of interconnected neurons arranged in layers. It is an ensemble method that learns complex relationships between input data and uses a layered structure to process the inputs and make a prediction [6]. In an ANN classifier, the weights of the connections between neurons are used to determine the class of a target species. ANN is similar to Random Forest in that it can handle high-dimensional datasets with many variables, but it is also able to capture the shape and complex relationships between the input layers. This makes it a powerful tool for classification tasks.

In the original paper, the author uses a model with only one single layer of hidden layers with 9 neurons. For the NDA type label, we got the training and validation accuracy and loss plots, overall scores, confusion matrix, and classification report respectively. And for the Kingdom label, we got the training and validation accuracy and loss plots, overall scores, confusion matrix, and classification report are shown below, respectively:

In summary, the results from the original paper show higher accuracy, and macro-F1 scores for both kingdom and DNA-type classification tasks compared to the results obtained in the current study. The accuracy for kingdom classification in the original paper was 0.9132, while the current study obtained a lower result of 0.8191. The micro-F1 score for kingdom classification in the original paper was 0.9546, while the current study obtained a lower result of 0.891. For DNA-type classification, the results from the original paper also showed higher accuracy, and micro-F1 scores compared to the current study. The accuracy in the original paper was 0.9997, while the current study obtained a slightly lower result of 0.9568. The micro-F1 score in the original paper was 0.9546, while the current study obtained a slightly higher result of 0.9568.

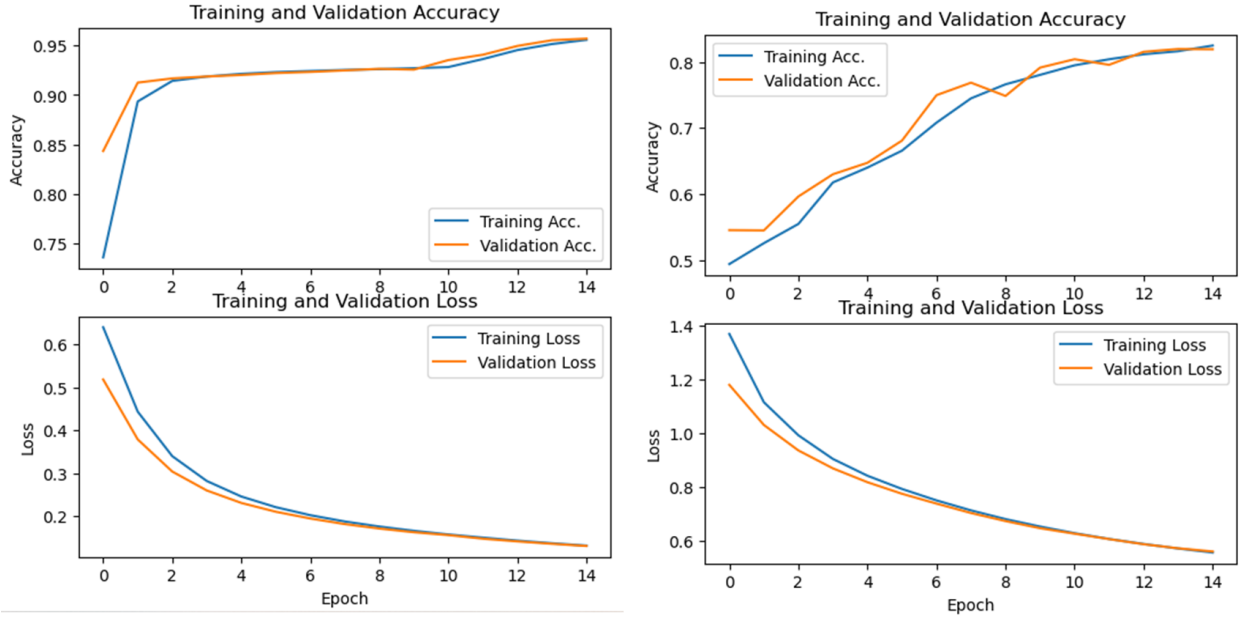


Figure 7. Training and validation accuracy and loss plots. Left: DNA type, Right: Kingdom.

```
precision_score:  0.9568067875048206   Confusion matrix :
recall_score:  0.9568067875048206       [[1851    1    0]
f1_micro_score:  0.9568067875048206     [  20  558    1]
accuracy_score:  0.9568067875048206     [  81    9   72]]
```

Classification report :

	precision	recall	f1-score	support
0	0.95	1.00	0.97	1852
1	0.98	0.96	0.97	579
2	0.99	0.44	0.61	162
accuracy			0.96	2593
macro avg	0.97	0.80	0.85	2593
weighted avg	0.96	0.96	0.95	2593

Figure 8. The classification report for the best ANN model for DNA type.

```
accuracy_score:  0.8191284226764366
precision_score:  0.7984150801335197
recall_score:  0.8191284226764366
f1_micro:  0.8191284226764366
f1 macro:  0.4845865000113805
```

Confusion matrix :

```
[[   0   10    2    0    9]
 [   0  491   33    0   20]
 [   0   70 1241    0  104]
 [   0   31   10    0    2]
 [   0   25  153    0  392]]
```

Classification report :

	precision	recall	f1-score	support
0	0.00	0.00	0.00	21
1	0.78	0.90	0.84	544
2	0.86	0.88	0.87	1415
3	0.00	0.00	0.00	43
4	0.74	0.69	0.71	570
accuracy			0.82	2593
macro avg	0.48	0.49	0.48	2593
weighted avg	0.80	0.82	0.81	2593

Figure 9. The classification report for the best ANN model for Kingdom.

Since the results are not good enough, we decided to have two hidden layers model with 128 and 32 neurons. For the kingdom label, we got the overall scores, confusion matrix, and classification report are shown below in Figure 10, respectively. And for the Kingdom label, we got the overall scores, confusion matrix, and classification report shown below in Figure 11, respectively.

```
accuracy_score:  0.9421519475510991
precision_score:  0.9434712800955324
recall_score:  0.9421519475510991
f1_micro:  0.9421519475510991
f1_macro:  0.8150553258026367
```

Confusion matrix :

```
[[  11    8    0    0    2]
 [   7  504   15   14    4]
 [   0   15 1346    1   53]
 [   0   11    0   32    0]
 [   2    1   17    0  550]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.55      0.52      0.54        21
           1       0.94      0.93      0.93       544
           2       0.98      0.95      0.96      1415
           3       0.68      0.74      0.71        43
           4       0.90      0.96      0.93       570

    accuracy                           0.94      2593
   macro avg       0.81      0.82      0.82      2593
weighted avg       0.94      0.94      0.94      2593
```

Figure 10. The classification report for the best optimized-ANN model for Kingdom.

```
accuracy_score:  0.9934438873891246
precision_score:  0.9934936554766272  Confusion matrix :
recall_score:  0.9934438873891246      [[1849    3    0]
f1_micro:  0.9934438873891246          [   2  577    0]
f1_macro:  0.9830435474959813          [   6    6  150]]


Classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1852
           1       0.98      1.00      0.99       579
           2       1.00      0.93      0.96       162

    accuracy                           0.99      2593
   macro avg       0.99      0.97      0.98      2593
weighted avg       0.99      0.99      0.99      2593
```

Figure 11. The classification report for the best optimized-ANN model for DNA type.

An optimized Artificial Neural Network (ANN) with multiple hidden layers performs better than an original ANN with only one single hidden layer in terms of accuracy and F1-micro score. This is because a deeper ANN with more hidden layers can learn more complex relationships between the input features and the output targets. However, it is important to carefully tune the architecture of the ANN, including the number of hidden layers and the number of neurons in each layer, to avoid overfitting or underfitting the data. It may also be helpful to use regularization techniques, such as dropout, to prevent overfitting. In addition, it is important to use an appropriate optimization algorithm and a suitable activation function for each layer. With the right data and careful tuning, an optimized ANN with multiple hidden layers can achieve higher accuracy and F1-micro scores than an original ANN with only one single hidden layer.

Naive Bayes Model

The technique of naive Bayes classifiers is based on the so-called Bayes' theorem and is especially suitable when the dimension of the input dimension is high [7]. In the formula below, P (class|X) refers to a posteriori probability that determines the classes sample data belongs. The numerator P (X|class) considers a sample belonging to the probability of a given class. The denominator P(class) at the bottom is the prior probability of the sample. The equation is shown below:

$$P(class|x) = \frac{P(x|class)P(class)}{P(x)}$$

The naive Bayes classifier by doing a conditional independence assumption significantly reduces the modeling needs to estimate the number of parameters. Therefore, the data associated with each feature is assumed to be distributed in a designed distribution. For both Kingdom and DNA type classifications, four different distributions are tested.

Table 1. Comparison of 5 Different Naive Bayes Models for Kingdom type

Model	Accuracy	Precision	Recall	F1-micro	F1-macro	AUC
BernoulliNB	0.7354	0.7095	0.7354	0.7354	0.4249	0.8880
GuassianNB	0.7343	0.7853	0.7343	0.7343	0.6149	0.9285
CompleteNB	0.6290	0.6768	0.6290	0.6290	0.3724	0.7790
MultinomialNB	0.6224	0.6175	0.6224	0.6224	0.2785	0.8760

Table 2. Comparison of 5 Different Naive Bayes Models for DNA type

Model	Accuracy	Precision	Recall	F1-micro	F1-macro	AUC
BernoulliNB	0.8897	0.8884	0.8897	0.8897	0.8196	0.9662
GuassianNB	0.9488	0.9488	0.9488	0.9488	0.8995	0.9854
CompleteNB	0.9250	0.9222	0.9260	0.9260	0.8192	0.9782
MultinomialNB	0.9171	0.9083	0.9171	0.9171	0.7432	0.9796

Regarding kingdom classifications in Table.1, the best Naive Bayes model yields an overall accuracy of 0.7343, and 0.6149 in the macro-F1 score which data is assumed to be Gaussian Distribution. On the other hand, the best model for the DNA type classification provides a relatively higher accuracy (0.9488) and macro-F1 score (0.8995) but similarly belongs to the Gaussian Distribution according to Table.2.

Compared to the results obtained from the paper, kingdom classification got overall accuracy of 0.7522, and 0.7032 in the macro-F1 score, and DNA type classification got overall accuracy of 0.9390, and 0.8910 in the macro-F1 score. Our results have a bit higher accuracy and macro-F1 score but it generally agrees with the conclusion from the paper.
The defect of Naive Bayes is that it is assumed that all predictive variables are independent in the sample data. However, this assumption is uncertain because we cannot assume complete codon frequency independence for biological reasons in this project [2]. Therefore, the Naive Bayes classifier is considered to be the least desirable solution to predict the taxonomic identity and genetic composition of biological species in this study.

Different Techniques

Logistic Regression

Logistic regression (LR) is based on the idea of using a linear model to estimate the log-odds of a binary outcome. The log-odds of an event is defined as the logarithm of the odds of the event occurring, where the odds of an event are defined as the probability of the event occurring divided by the probability of the event not occurring [8]. The function of logistic regression is shown below:

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

where u is a location parameter and s is a scale parameter [9]. For DNA type label, we got the overall scores, confusion matrix, and classification report are shown below in Figure 12, respectively. For the Kingdom label, we got the overall scores, confusion matrix, and classification report are shown below in Figure 13, respectively.

```
accuracy_score:  0.9845738526802931
precision_score:  0.9844413365780911
recall_score:  0.9845738526802931
f1_micro:  0.9845738526802931
f1_macro:  0.9636795377520104
auc_score:  0.9983212039223415
```

```
Confusion matrix :
 [[1846    3    3]
 [  12  564    3]
 [  14    5  143]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1852
           1       0.99      0.97      0.98       579
           2       0.96      0.88      0.92       162

    accuracy                           0.98      2593
   macro avg       0.98      0.95      0.96      2593
weighted avg       0.98      0.98      0.98      2593
```

Figure 12. The classification report for the best optimized-ANN model for DNA type.

```
accuracy_score:  0.802159660624759
precision_score:  0.8013312112611557
recall_score:  0.802159660624759
f1_micro:  0.802159660624759
f1_macro:  0.5739514559202237
auc_score:  0.9417674974826051
```

```
Confusion matrix :
 [[   3   10    6    0    2]
 [   4  490   32    0   18]
 [   2   74 1208    0  131]
 [   0   23   10    8    2]
 [   2   25  172    0  371]]
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.27      0.14      0.19        21
           1       0.79      0.90      0.84       544
           2       0.85      0.85      0.85      1415
           3       1.00      0.19      0.31        43
           4       0.71      0.65      0.68       570

    accuracy                           0.80      2593
   macro avg       0.72      0.55      0.57      2593
weighted avg       0.80      0.80      0.80      2593
```

Figure 13. The classification report for the best optimized-ANN model for Kingdom.

The overall scores of either Kingdom or DNAtype labels are lower than the model used by the author of the original paper. Hence, logistic regression is not suitable for this dataset.

Decision Tree

The decision tree is a Tree-like model usually used for making decisions and classifications. As an algorithm we used, it is a non-parametric algorithm for supervised learning. [10] It starts with a node called the root node, as the top beginning level. The root node would lead to other nodes called internal nodes. When there is no better way to specify further or classify our data, the options will end at the point we call them "leaves."

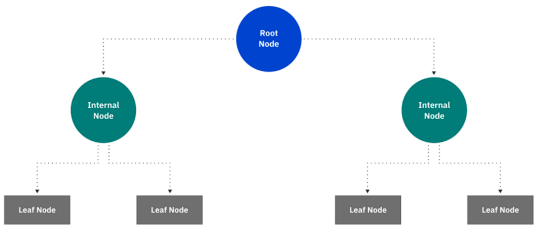


Figure 14. a simple plot of the Decision tree. [10]

There are two main types of values to help classify more precisely: entropy, and Gini impurity. Entropy measures the impurity of the information values, so we want it to be as small as possible when doing classification. Entropy cannot be bigger than 1. Gini impurity reflects the probability of incorrect classified data in the dataset. To make our tree precise, we must also make Gini impurity as low as possible. In our data, we followed the steps and rules above. Finally, we built

our decision tree for a classification tree. However, a decision tree combines some decisions, whereas a random forest combines several decision trees. The accuracy of the decision tree should be less than RF gained. According to the comparison table in Table 3 and 4, our inference is in line with the actual situation. Therefore, the decision tree is also considered to be the least desirable solution to predict the result of this project.

Result and Discussion

A comparison of the AUC (area under curve) value between the six aforementioned classifiers (k-NN, RF, XGB, ANN, LR, and Decision Tree) is presented below. The naive Bayes classifier is omitted because it violates the independence assumption. The area under the ROC (receiver operating characteristic) curve is measured as the AUC value under each chart. These ROC curves describe the tradeoff between TPR(true positive rate) and FPR(false positive rate) for the selected classifier. The ROC curve closer to the upper left classifier indicates a good fit and a high AUC value (closer to 1).

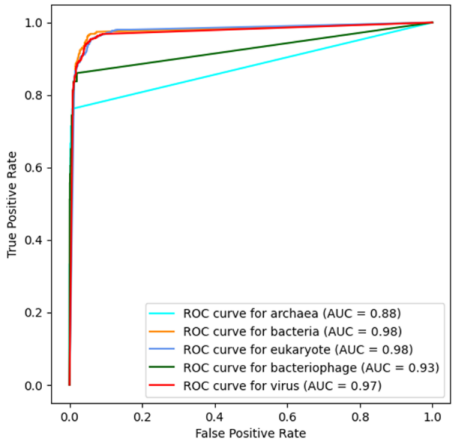
In Figure 15, the ROC curves are compared for kingdom classification analysis. Both the Random Forest and XGBoost classifier show smooth ROC curves closer to 1. This means that both models are good predictors of kingdom types. The ROC curves in the 4-Nearest Neighbor and ANN classifier not only give a smaller y-axis value (TPR) indicating lower true positives and higher false negatives but also not uniformly across the five kingdoms. For kingdom archaea and bacteriophage, the AUC values for both k-NN and ANN classifiers are significantly lower than other kingdom classes.

In Figure 16, the ROC curves are compared for the DNA type classification analysis. All five ROC curves except the Decision Tree are similarly near the upper left corner, indicating that the AUC values of the four classifiers are close to 1. This indicates that these five machine learning classifiers have good model fitting and performance in predicting DNA types of species.

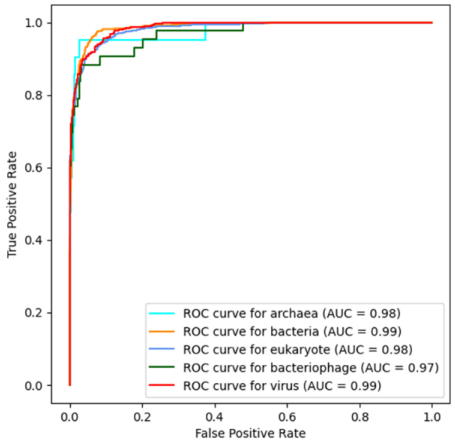
Table 3 and Table 4 give detailed performance metrics of the four classifiers for Kingdom and DNA type classifications respectively. In Table 3, although the accuracy (0.9434) and macro-f1 score (0.8151) of the optimized ANN model is the highest, the AUC value (0.9120) is lower than RF (0.9786) and XGBoost (0.9744). Therefore, RF and XGBoost have more capability to distinguish between correct and incorrect classes for the samples for kingdom classification. According to table 4, all five classifiers except Naive Bayes and Decision Tree yield high precision and overall accuracy (> 0.98) values. Therefore, these five classifiers are all good choices for predicting the genetic composition of selected organisms based on codon usage frequency in DNA classification analysis

Figure 17 shows clusters of samples based on their similarity by various dimensionality reduction methods. The left three panels represent DNA type and the right three panels represent Kingdom. There is not much separation observed in Kingdoms. The five main classes are merged together and no clear separation boundaries. That means it is difficult to distinguish the kingdoms according to the frequency of codon use. However, the DNA type shows a more clear separation for the three main classes. Each class is highly homogeneous and distinct from the

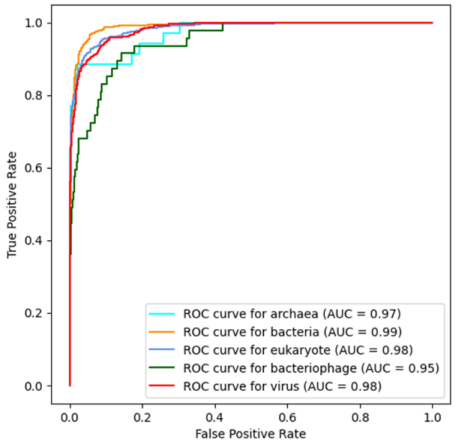
others. The separation line was good for separating these points while producing three other distinct clusters. This is the reason why the overall accuracy and precision of classifiers for the kingdom are lower than DNA type.



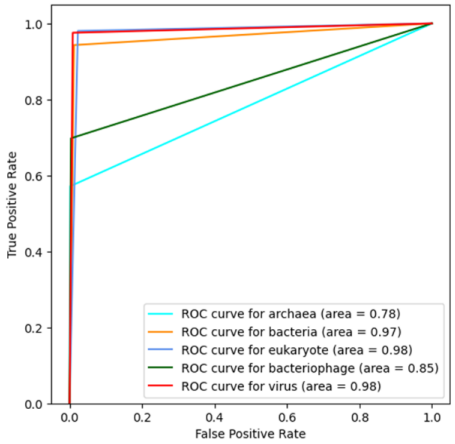
(a) k-Nearest Neighbor (k=4) model (AUC=0.9468)



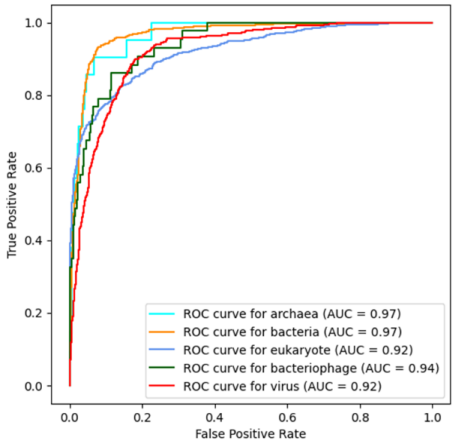
(b) Random Forest model (AUC=0.9786)



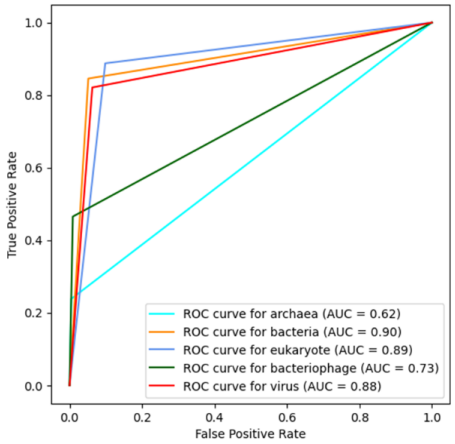
(c) XGBoost model (AUC=0.9744)



(d) Artifificial Neural Network model (AUC=0.9120)

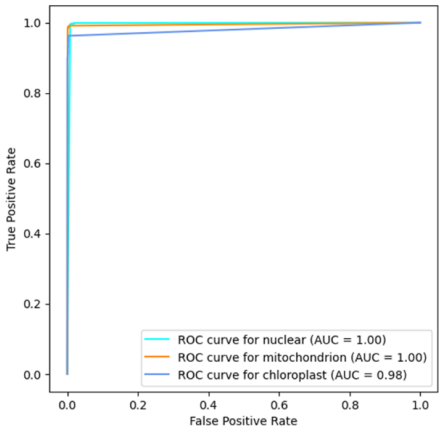


(e) Logistic Regression (AUC=0.9418)

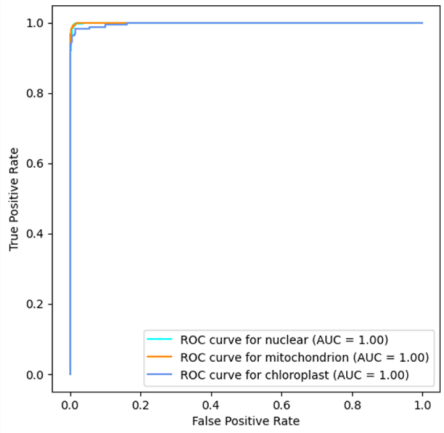


(f) Decision Tree (AUC=0.7981)

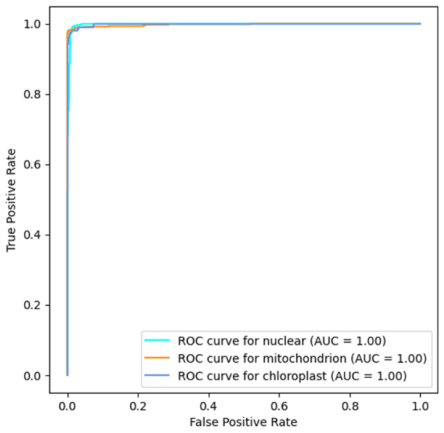
Figure 15. AUC ROC curves for the kingdom classification use 6 different methods.



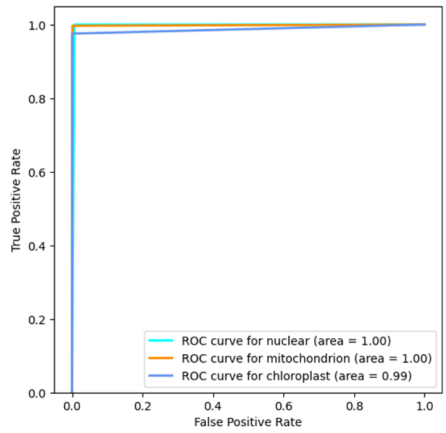
(a) k-Nearest Neighbor (k=4) model (AUC=0.9906)



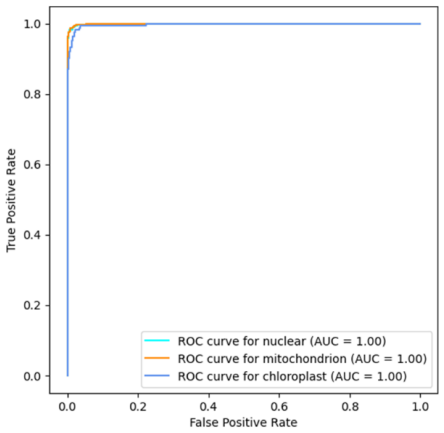
(b) Random Forest model (AUC=0.9987)



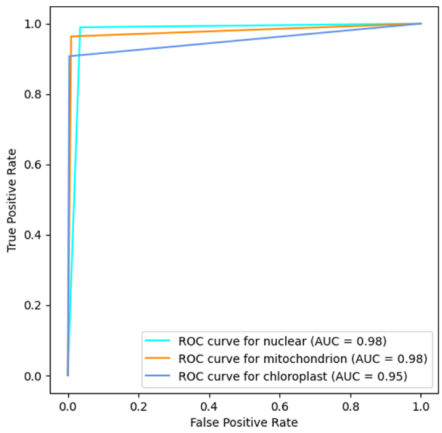
(c) XG-Boost model (AUC=0.9979)



(d) Artifificial Neural Network model (AUC=0.9967)



(e) Logistic Regression (AUC=0.9983)



(f) Decision Tree (AUC=0.9620)

Figure 16. AUC ROC curves for the DNA type classification use 6 different methods.

Table 3. Comparison of 7 different selected models for kingdom classification

Model	Precision	Recall	Micro-F1 Score	Macro F1 Score	Accuracy	AUC
k-NN	0.9225	0.9229	0.9229	0.8063	0.9229	0.9468
Random Forests	0.9027	0.9028	0.9028	0.7048	0.9028	0.9786
XGBoost	0.9060	0.9071	0.9071	0.7715	0.9071	0.9744
ANN-Optimized	0.9434	0.9421	0.9421	0.8151	0.9421	0.9120
Naive Bayes	0.7343	0.7853	0.7343	0.7343	0.6149	0.9285
Logistic Regression	0.8013	0.8022	0.8022	0.5740	0.8021	0.9418
Decision Tree	0.8523	0.8511	0.8511	0.6427	0.8511	0.7981

Table 4. Comparison of 7 different selected models for DNA type classification

Model	Precision	Recall	Micro-F1 Score	Macro-F1 Score	Accuracy	AUC
k-NN	0.9919	0.9919	0.9919	0.9788	0.9919	0.9906
Random Forests	0.9846	0.9846	0.9846	0.9655	0.9846	0.9987
XGBoost	0.9873	0.9873	0.9873	0.9725	0.9873	0.9979
ANN-Optimized	0.9935	0.9934	0.9934	0.9830	0.9934	0.9967
Naive Bayes	0.9488	0.9488	0.9488	0.9488	0.8995	0.9854
Logistic Regression	0.9844	0.9846	0.9846	0.9637	0.9846	0.9983
Decision Tree	0.9741	0.9742	0.9742	0.9399	0.9742	0.9620

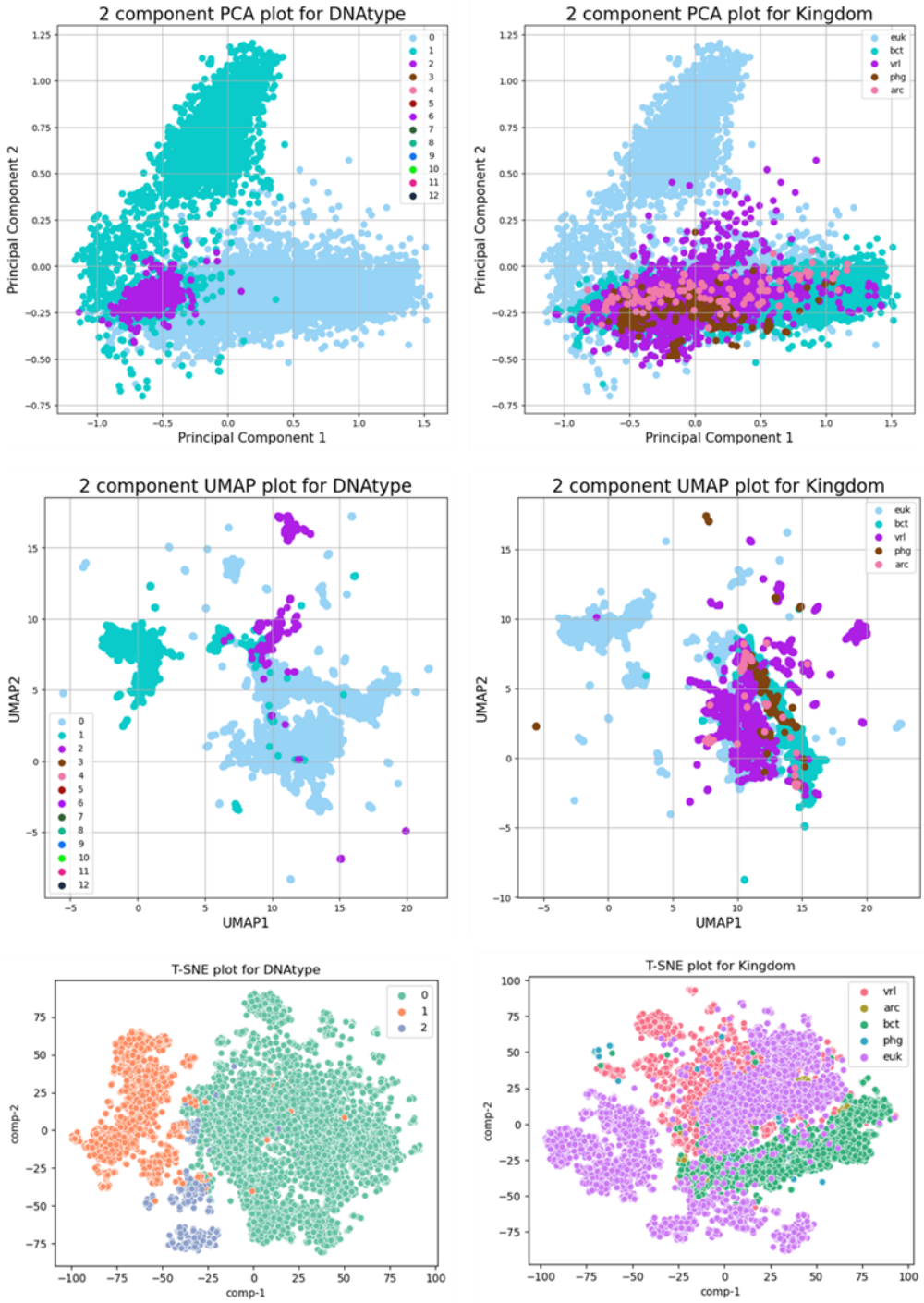


Figure 17. UMAP, t-SNE, and PCA plots for Kingdoms and DNAType. Left: DNA type, Right: kingdoms.

Conclusion

Looking at the model performance presented by the seven machine learning classifiers, our analysis shows that the k-NN, XGBoost, and RF algorithms are the three best solutions for

classifying kingdom classes in the secondary analysis of existing genetic data sets. Although optimized ann classifiers exhibited the highest accuracy (0.9434) and overall accuracy (0.9421) of model performance, XGBoost and RF classifiers produced higher AUCs (0.9744 and 0.9786, respectively) for better model fit in their ROC curves. For the prediction of biological DNA types, all five classifiers (k-NN, XGB, RF, optimized-Ann, and LR) showed high precision (> 0.98) of model performance and overall accuracy (> 0.98), optimized-ANN showed high micro and macro f1 scores. In particular, the ROC curves of XGB and RF in the domain (AUC > 0.97) and DNA type (AUC > 0.99) showed the best performance and model fitting in all categories. Thus, extreme gradient boost and random forest algorithms can be used to build the most robust and efficient classifiers to identify and predict taxonomic and genetic characteristics of organisms. According to the results of choose paper, we figure out that the conclusion of best classification models is similar.

Future Work

Codon frequency is a shortcut to DNA classification. These codon usage frequencies are not only useful features for predicting the taxonomic identity of living organisms but also powerful tools for classifying the genetic composition of life forms [2]. Further development in the future should address the independence of different specie. Although codon frequencies belong to different species, there are still correlations between independent species. If we can solve this correlation problem, the accuracy of prediction can be improved further on.

Reference

[1] “UCI Machine Learning Repository: Codon usage Data Set,” Uci.edu. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Codon+usage. [Accessed: 13-Dec-2022]

[2] Bohdan B. Khomtchouk. “Codon Usage Bias Levels Predict Taxonomic Identity and Genetic Composition” University of Chicago Department of Medicine, Oct 27, 2020

[3] Srivastava, Tavish. “K Nearest Neighbor: Knn Algorithm: KNN in Python & R.” *Analytics Vidhya*, Oct 18, 2020, https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.

[4] “Random Forest.” *Random Forest - an Overview | ScienceDirect Topics*, https://www.sciencedirect.com/topics/engineering/random-forest.

[5] Amazon.com. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html. [Accessed: 13-Dec-2022]

[6] IBM Cloud Education. “What Are Neural Networks?” *IBM*, https://www.ibm.com/cloud/learn/neural-networks.

[7] Vijaykumar B, Vikramkumar, Trilochan. “Bayes and Naive-Bayes Classifier”. Computer Science & Engineering Rajiv Gandhi University of Knowledge Technologies Andhra Pradesh, India. https://arxiv.org/ftp/arxiv/papers/1404/1404.0933.pdf

[8] “What Is Logistic Regression?” *IBM*, https://www.ibm.com/topics/logistic-regression.

[9] “Logistic Regression.” *Wikipedia*, Wikimedia Foundation, 19 Dec. 2022, https://en.wikipedia.org/wiki/Logistic_regression.

[10] “What is a Decision Tree?” *IBM*, https://www.ibm.com/topics/decision-trees