CODON USAGE BIAS TO CLASSIFY ORGANISM'S KINGDOM CLASS CLASSIFICATION USING SUPERVISED AND UNSUPERVISED CLUSTERING

Adarsha Bhuvana Chandran*
Nishit Chaudhry*
Pranav Pailkar*
University at Buffalo
Buffalo, NY 14260

Abstract: *In this study, supervised and unsupervised machine learning algorithms along with various steps involved in building a machine learning model were studied. For this genetic data of organisms was used as a classification problem in which the goal was to classify organisms into their respective Kingdom class (Taxonomy Identification) from codon usage frequencies in organism's genomic features. We tried to achieve this result using a few supervised and unsupervised algorithms using available genomic data. AI model was trained to classify a total of five Kingdom classes namely Plants, Bacteria Invertebrates, Viruses, and Vertebrates. Over 9000 species samples were used to train AI models and for clustering in this study. In total from the genetic coding of the organism 64, codon usage frequencies were used as features. After analyzing the results from the supervised models and clusters formed in the unsupervised technique our findings imply that these codon usage patterns can be reliably used for predicting the class of the model.*

1. Introduction

The coding DNA of a genome describes the proteins of the organism in terms of 64 different codons that map to 21 different amino acids and a stop signal. Different organisms differ not only in the amino acid sequences of their proteins but also in the extent to which they use the synonymous codons for different amino acids. The inherent redundancy of the genetic code allows the same amino acid to be specified by one to five different codons so that there are, in principle, many different nucleic acids to describe the primary structure of a given protein. Coding DNA sequences thus can carry information beyond that needed for encoding amino acid sequence [1].

In this study, we are trying to classify codon usage to predict and segregate the Kingdom Class of organisms from 64 codons and we demonstrated that different genomic and evolutionary features can be learned using machine learning methods. However, these machine learning algorithms, such as k-Nearest Neighbors (k-NN) [2], Random Forests (RF) [3], and Super Vector Machine (SVM), each demonstrates unique model performances with different degrees of efficiency and accuracy. Therefore, we set out to compare and benchmark these algorithms with that unsupervised clustering algorithms.

2. Dataset

2.1 Data Description

The codon usage consists of codon usage frequencies in the genomic coding DNA of a large sample of diverse organisms from different taxa tabulated in the CUTG database.

We have several attributes such as Kingdom, DNAtype, SpeciesID, Ncodons, SpeciesName and codon frequencies. The 'Kingdom' is a 3-letter code corresponding to `xxx' in the CUTG database name: 'arc' (archaea), 'bct' (bacteria), 'phg' (bacteriophage), 'plm' (plasmid), 'pln' (plant), 'inv' (invertebrate), 'vrt' (vertebrate), 'mam' (mammal), 'rod' (rodent), 'pri' (primate), and 'vrl' (virus) sequence entries.

The DNAtype is denoted as an integer for the genomic composition in the species: 0-genomic, 1-mitochondrial, 2-chloroplast, 3-cyanelle, 4-plastid, 5-nucleomorph, 6-secondary_endosymbiont, 7-chromoplast, 8-leucoplast, 9-NA, 10-proplastid, 11-apicoplastic and 12-kinetoplast.

The species identifier ('SpeciesID') is an integer, which uniquely indicates the entries of an organism. It is an accession identifier for each different species in the original CUTG database, followed by the first item listed in each genome.

The number of codons (`Ncodons') is the algebraic sum of the numbers listed for the different codons in an entry of CUTG. Codon frequencies are normalized to the total codon count, hence the number of occurrences divided by 'Ncodons' is the codon frequencies listed in the data file.

The species' name ('SpeciesName') is represented in strings purged of `comma' (which are now replaced by `space'). This is a descriptive label of the name of the species for data interpretations.

2.2 Data Collection

The dataset was taken from the UCI machine learning repository. Bohdan Khomtchouk, Ph.D. The University of Chicago, Department of Medicine, Section of Computational Biomedicine and Biomedical Data Science owns it.

Date Donate: 2020-10-03

License:

Khomtchouk BB: 'Codon usage bias levels predict the taxonomic identity and genetic composition'. bioRxiv, 2020, doi: 10.1101/2020.10.26.356295.

Nakamura Y, Gojobori T, Ikemura T: 'Codon usage tabulated from international DNA sequence databases: status for the year 2000'. Nucleic Acids Research, 2000, 28:292.

2.3 Data Classes

The dataset has 11 classes namely:

1. vrl for virus,
2. bct for bacteria,
3. pln for plants,
4. inv for invertebrates,
5. vrt for vertebrates,
6. arc for archaea,
7. phg for bacteriophages,
8. plm for plasmids,
9. mam for mammals,
10. rod for rodents,
11. pri for primates

Based on the unique Kingdoms and there are a total of 13028 observations.

2.4 Data Distribution

CLASS	OBSERVATIONS
Virus	2832
Bacteria	2920
Plants	2523
Invertebrates	1345
Vertebrates	2077
Archaea	126
Plasmids	18
Primates	180
Rodents	215
Mammals	572
Bacteriophage	220

2.5 Data Exploration

The 64 features from the dataset were analyzed by plotting 2D and 3D plots. Out of which some features were found to be separating data up to some extent.

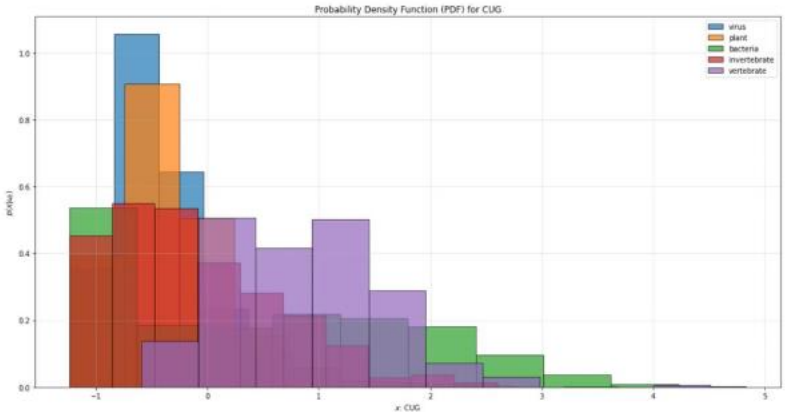


Fig 1: Histogram plot for Codon usage - 'CUG'

The plot illustrates the class labels distribution namely Virus, Plants, Bacteria, Invertebrate, and Vertebrate for the input feature Codon usage – 'CUG'

All the Codon usage frequencies (63) were plotted one by one for 5 class labels. Hence, examining of features was done randomly in this case. The separation between the class labels was evident here as seen for the kingdom type 'Vertebrate' in comparison to other codon usage frequencies plots

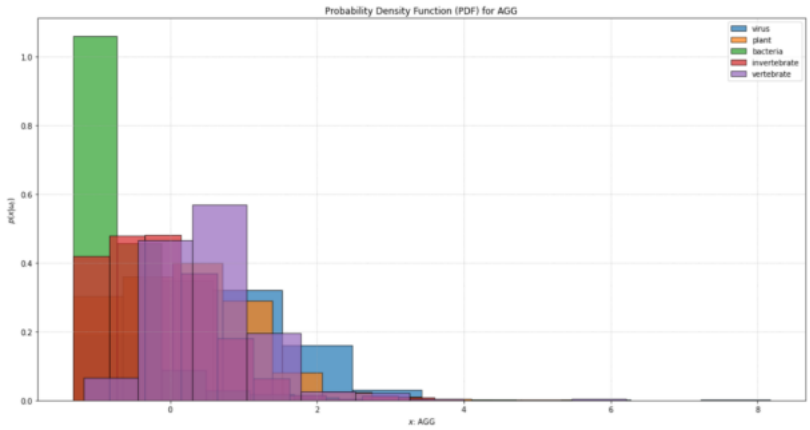


Fig 2: Histogram plot for Codon usage - 'AGG'

The above plot illustrates the class label distribution for codon usage 'AGG'. The plot exhibits different class distribution where the classes are more clustered together

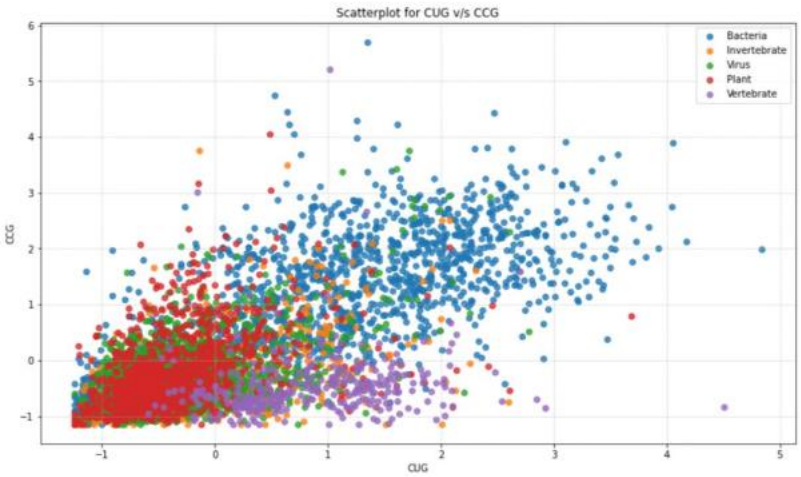


Fig 3

3. Methodology

In this study, our goal is to examine the performance of supervised and unsupervised machine learning algorithms to predict and segregate the Kingdom Class of organisms from 64 codon usage bias level [4]. This classification was achieved through two techniques: 1) Using 64 codon usage frequencies as features to fit supervised machine learning algorithms like K-Nearest Neighbors, Support Vector Machines, and Random Forest. 2) Unsupervised clustering to check if each class were able to form a cluster or not using algorithms like K-Means, Agglomerative, and Spectral Clustering. As for this study out of a total of 11 kingdom classes top 5 classes which had more samples than other were selected.

3.1 Supervised Learning

K-Nearest Neighbor, Random Forest, and Support Vector Machines were the classifiers used on this dataset. As seen from data plots in section 2.5, features are arranged in clusters so classifying new data points having reference of training sample in features can give better results. And since this dataset has multiple classes, it is more suited to use KNN and random forest, but we saw better results on the SVM model.

Iterations performed with different algorithms:

•KNN and SVM models were used on all features together

•Random Forest was implemented on the top 30 features

•SVM was implemented on important features selected from the data plotting assignment

A train-test split of 70:30 on a total of 8657 samples by stratifying the classes.

```
====================
 Train-Test Split
====================

Overall class ratio:
0    0.336953
3    0.327019
2    0.175927
1    0.106503
4    0.053598
Name: Kingdom, dtype: float64

Train set class ratio:
0    0.337019
3    0.326952
2    0.175937
1    0.106453
4    0.053639
Name: Kingdom, dtype: float64

Test set class ratio:
0    0.336798
3    0.327175
2    0.175905
1    0.106620
4    0.053503
Name: Kingdom, dtype: float64
```

Once the model is trained analysis to detect outliers is performed. For outlier detection records having values greater than 3 standard deviations were outliers. Out of 193 incorrect predictions from a total of 2165 predictions, 77 records were outliers if 3 standard deviation criteria are followed. Records are marked outliers if any feature out of 64 is an outlier with the above criteria.

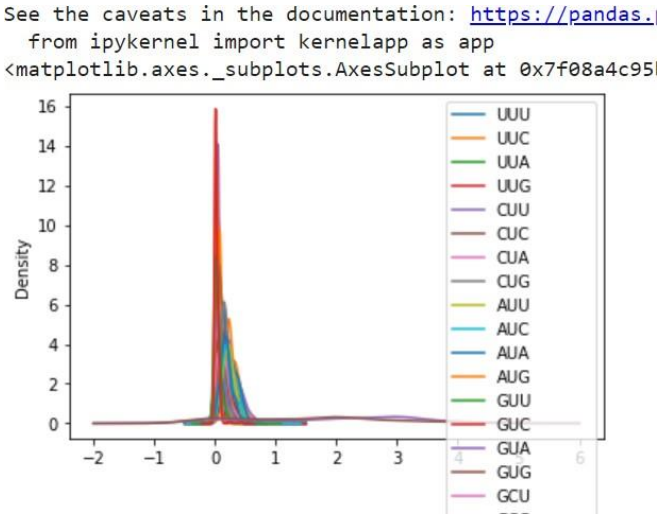


Fig 4: Probability Density Plot of features on KNN model

Performance Optimization

Method 1: Feature Selection by importance

The top 30 important features from the base model were taken as a subset to train to check if any performance difference is observed. Important features were taken from feature_importances_function in scikit-learn. The model trained on these 30 features had a reduction of only 0.6%, which isgood considering only half of the original features were selected.

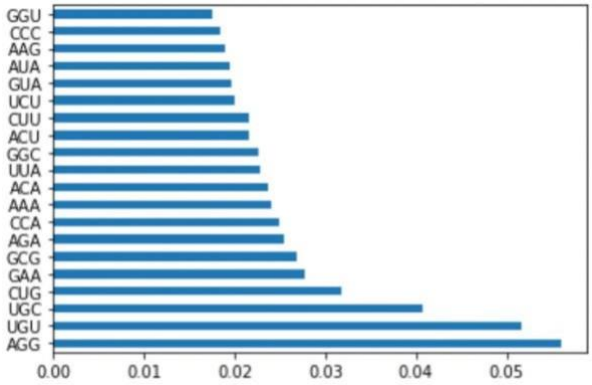


Fig 5: Displaying Top 20 features from the base KNN model

Further on these 30 features, k-fold validation was run with k=10 as suggested in the assignment. Which gave the average accuracy of 88.82%

Method 1: Feature Selection by Plots

In this method, we chose the features that were highlighted in our data plotting analysis. The SVMmethod with the same set of parameters for GridSearchCV was used here to get the below results.

The selected features from data plotting:

['AGA', 'AGG', 'CCG', 'CUG', 'GGC', 'GGG', 'GCA', 'UGG', 'GUA', 'UAG', 'CAC', 'AUU', 'AUG']

```
svm_train_acc = grid_cvv.score(train_sel_features_scaled, train_sel_labels)
print(f"Support Vector Machine Accuracy (Training): {svm_train_acc:.4}")

svm_test_acc = grid_cvv.score(test_sel_features_scaled, test_sel_labels)
print(f"Support Vector Machine Accuracy (Testing): {svm_test_acc:.4}")
```

```
Support Vector Machine Accuracy (Training): 0.9089
Support Vector Machine Accuracy (Testing): 0.8337
```

```
print(55 * "=")
print("Support Vector Machines")
print(55 * "-")
print(classification_report(test_sel_labels, predict_sel_labels))
print(55 * "=")
```

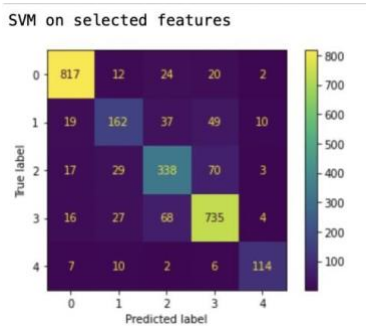
```
=======================================================
Support Vector Machines
-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       875
           1       0.68      0.58      0.63       277
           2       0.72      0.74      0.73       457
           3       0.84      0.86      0.85       850
           4       0.86      0.82      0.84       139

    accuracy                           0.83      2598
   macro avg       0.80      0.79      0.80      2598
weighted avg       0.83      0.83      0.83      2598

=======================================================
```

Fig 6: Results



This method gives a lower performance which means that other features are also important in explaining the classes, but we also noticed that the performance was lower for some classes (especially1) and if we further add more features from here onwards, we could increase the model performance.

3.2 Unsupervised Learning

Before applying unsupervised learning to this dataset initial plots of the features were analyzed. For data visualization, we plotted 1-D, 2-D, and 3-D for both “selected” as well as “all features” and that was done using dimensionality reduction. Some of the obtained plots are shown below:

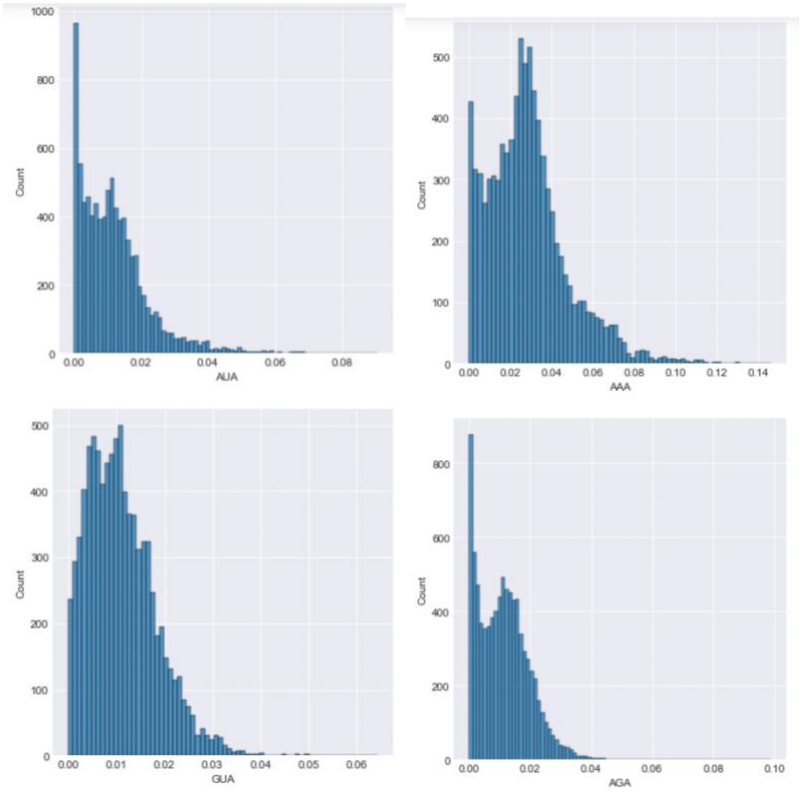


Fig 7: Histograms (1-D plots) of selected features

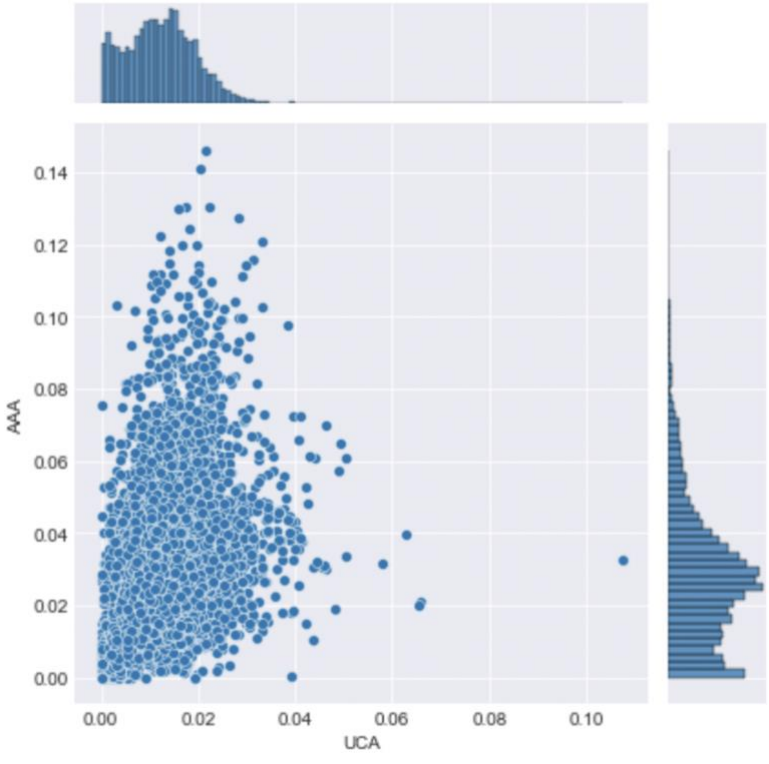


Fig 8: Scatter Plots (2-D Plots) on Selected Features

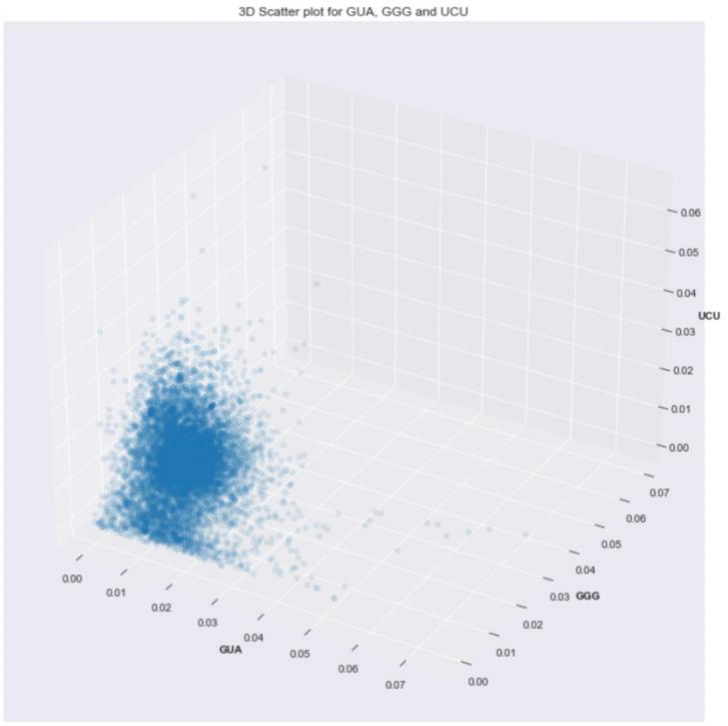
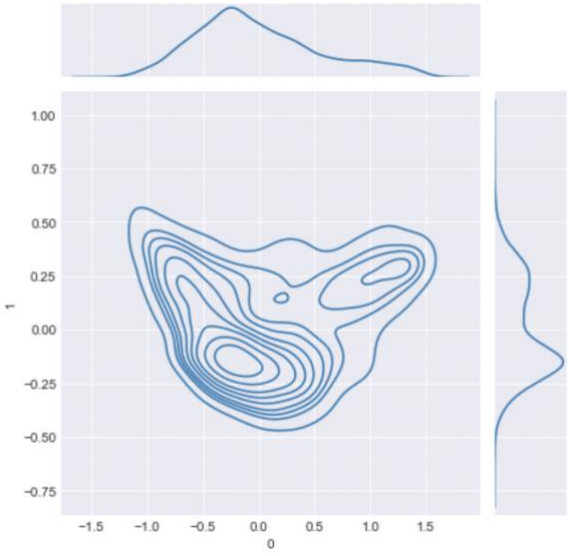


Fig 9: From 3D plots using features ('GUA', 'GGG', 'UCU'):

We observed from the selected features ('GUA', 'GGG', 'UCU') that 2-3 clusters were identifiable based on the 3-D plot.

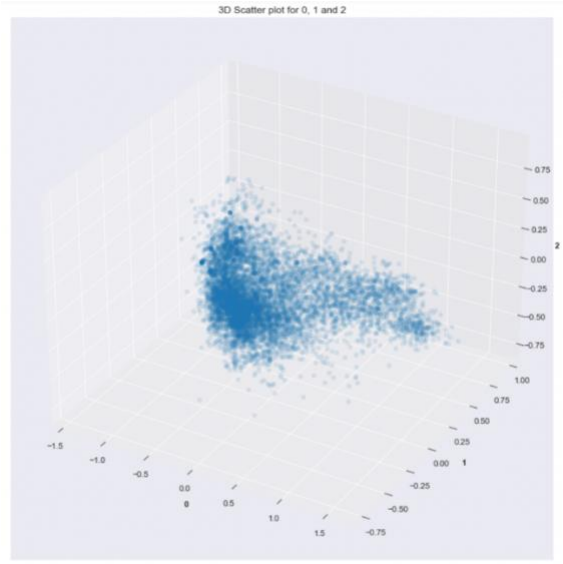
Using Dimensional reduction on all features and observing a 2-D plot:

On using dimensionality reduction on all the features and using 2 components in PCA, we observed that 2 clusters were visible.



Using Dimensionality reduction on all features and observing a 3-D plot:

On using dimensionality reduction on all the features and using 3 components in PCA, we observed that 3-4 clusters were visible.



From our analysis, we observed that clusters seem close together and we could detect clusters at most (4) but the clusters were overlapping. After using Principal Component Analysis, they seem to provide a clue that the dataset contains 4 classes whereas the original dataset contains 5 classes. K-means is used as the first clustering algorithm in this study. Parameters used where n_clusters, init as "k-means++", max_iter = 300, n_init = 10, and random_state = 5. n_clusters clusters were checked for values 1 to 100. It was found that at n_clusters = 4 the performance does not change significantly for the K-means model where all predictors are selected. From these results, it signifies that the model suggests fewer clusters than the expected no of 5 clusters from the dataset labels for the kingdom. Because the original dataset had 5 labels from 0 to 4.

4. Results

4.1 Supervised Learning

The performance of the models is measured on the below-mentioned metrics:

- Accuracy
- AUC
- F1-score (specifically, 'f1_micro' since it's a better strategy in case of imbalanced classes)
- Precision
- Recall
- Specificity

We have used the classification report and plot confusion matrix method in scikit-learn as well ascalculated AUC and Specificity scores.

Classification Reports

```
=====================================================
K Nearest Neighbors
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       850
           1       0.95      0.97      0.96       875
           2       0.85      0.89      0.87       457
           3       0.84      0.66      0.74       277
           4       0.87      0.91      0.89       139

    accuracy                           0.91      2598
   macro avg       0.89      0.88      0.88      2598
weighted avg       0.91      0.91      0.91      2598

=====================================================

=====================================================
Random Forest on top 30 features
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       850
           1       0.95      0.98      0.96       875
           2       0.87      0.84      0.86       457
           3       0.87      0.63      0.73       277
           4       0.91      0.81      0.85       139

    accuracy                           0.90      2598
   macro avg       0.90      0.84      0.86      2598
weighted avg       0.90      0.90      0.90      2598

=====================================================
```

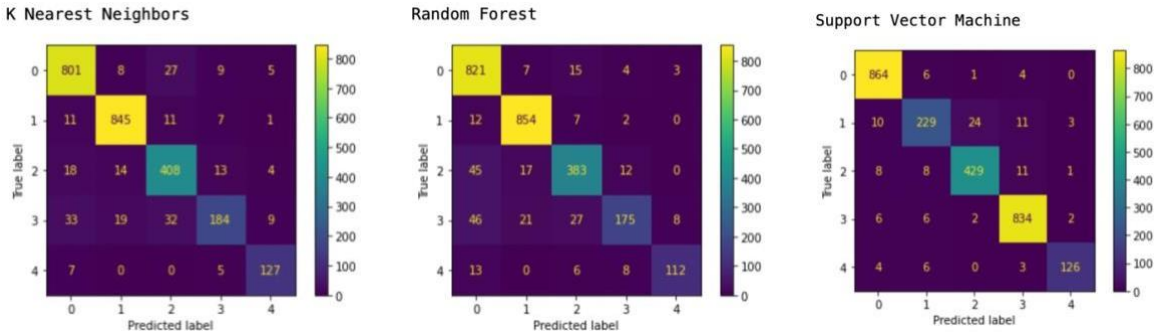
```
=====================================================
Support Vector Machines
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       875
           1       0.90      0.83      0.86       277
           2       0.94      0.94      0.94       457
           3       0.97      0.98      0.97       850
           4       0.95      0.91      0.93       139

    accuracy                           0.96      2598
   macro avg       0.95      0.93      0.94      2598
weighted avg       0.95      0.96      0.95      2598

=====================================================
```

Confusion Matrix



Calculated Scores:

	KNN	Random Forest	SVM
Accuracy	91.08	90.44	95.54
AUC	0.97	0.98	
Specificity (class 0-4)	0.96 0.97 0.96 0.98 0.99	0.94 0.97 0.97 0.98 0.99	0.98 0.98 0.98 0.98 0.99
Note	All features are used	Top 30 features are used	All features are used

Each model seems to perform better in some metrics as compared to others. So having an ensemble of these models might give a better performance. For example, the accuracy of KNN is better than RF but Area under the curve for RF is greater than KNN. The same patterns can seem in class-wise accuracy of both models. Whereas the SVM model gives a good overall performance on different metrics.

Model Performance compared to other studies

Our model performance is comparable with other studies done in past. Researchers have achieved anaccuracy of over 96% from KNN, whereas the accuracy in this study is 91% from our KNN. This is primarily because, in this study, only primitive preprocessing steps and optimization techniques are used.

However, the SVM model achieved an accuracy of 95.54% with the below-mentioned input parameterson a k-fold CV of 10:

SVM Parameters

```
grid_cv.best_params_
```

```
{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
```

4.2 Unsupervised Clustering

	model	calinski harabasz score	davies bouldin score	silhouette score
0	kmeans	2268.487281	2.157803	0.119871
1	spectral	1478.843625	2.549360	0.090372
2	aggloclust	2029.603703	1.910959	0.127523

Calinski-Harabasz score: The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

Davies Bouldin score: Zero is the lowest possible score. Values closer to zero indicate a better partition.

Silhouette Score: The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

Comparison between unsupervised algorithms used from sklearn package. All clustering algorithms available in the sklearn cluster package were tried on this codon dataset. Out of which only three converged of whom the scores are calculated. K-means and Agglomerative Clustering gave a better Calinski-Harabaz score on the dataset as compared to the Spectral clustering algorithm.
As for the Silhouette Index similar performance is observed among these algorithms. Agglomerative Clustering has a higher score than the other two algorithms. All these models are trained to give out 5 clusters since the number of levels in the original dataset is three. This score seems to deteriorate as the number of clusters asked increases from 2 and above.

5. Discussion

Codon usage analyses have great potential in revealing the genetic evolution and mutation causes of living species. Future research on codon usage analyses might extend the explanation to phylogenetic lineages using a subset of characteristic codon frequencies when examining species that share similar genetic features. Furthermore, future work might apply a combined machine learning strategy, such as combining Principal Component Analysis (PCA) and other machine learning classifiers to study codon usage bias levels with a reduced codon frequency dimension.

6. Conclusion

In the supervised classification techniques, we were able to classify the samples into 5 classes as predicted, with a minimum accuracy of better than 90%, implying that we properly identified 90% of the samples. However, in unsupervised clustering algorithms, the best we could achieve was just four clusters, and even then, the clusters overlapped. As a result, the unsupervised classification results were not as effective as the supervised classification results, and we are more confident in the supervised classification findings.

7. References

1. Khomtchouk, B.B., *CODON USAGE BIAS LEVELS PREDICT TAXONOMIC IDENTITY AND GENETIC COMPOSITION.* bioRxiv, 2020.
2. Lantz, B., *Machine Learning With R. Lazy Learning - Classification Using Nearest Neighbors.* In Packet Publishing, 2015: p. 65–86.
3. Zurab Bzhalava, A.T., Piotr Bała, Raul Vicente and Joakim Dillner, *Machine Learning for detection of viral sequences in human metagenomic datasets.* BMC Bioinformatics, 2018.
4. Nakamura Y, G.T., Ikemura T, *Codon usage tabulated from international DNA sequence databases.* Nucleic Acids Research, 2000.