

Predicting Kingdom

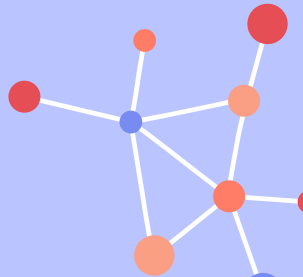
Team Murphy





Problem Statement

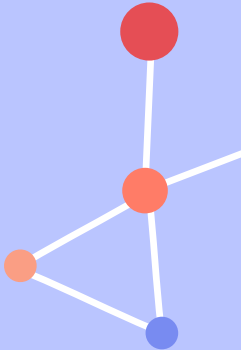
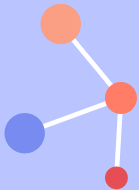
Predict the kingdom to which an observation belongs based on the data provided



Description of Data Set

The data set consisted of information about different species that belong to different Kingdoms and also elaborate information about different codons in each species. Details:

- 1) Column 1: Kingdom
- 2) Column 2: DNAType
- 3) Column 3: SpeciesID
- 4) Column 4: Ncodons
- 5) Column 5: SpeciesName
- 6) Columns 6-69: codon (header: nucleotide bases; entries: frequency of usage (5 digit floating point number))



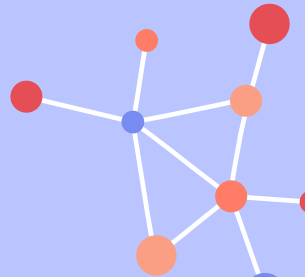


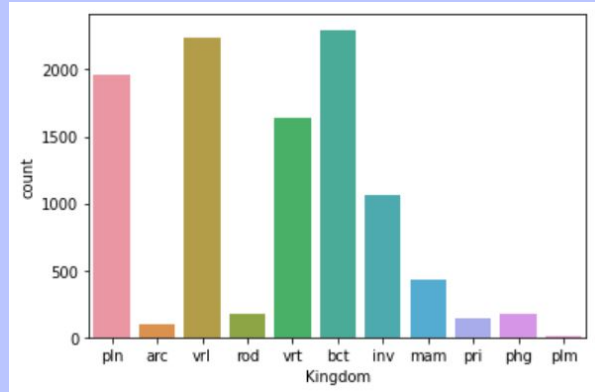
Data Exploration



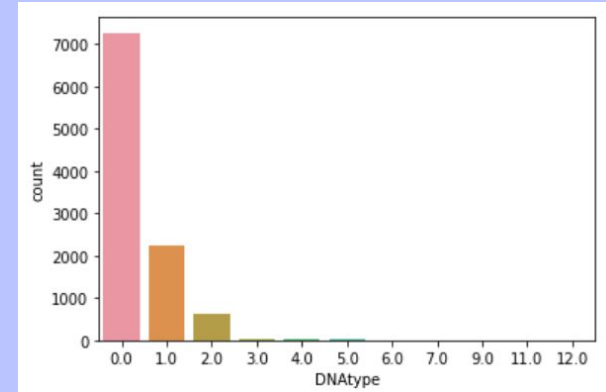
Data exploration highlights the traits of a single variable and also reveals patterns and relationships between variables

We use this to identify variables, identify outliers, to perform variable transformation and creation, and to find missing values

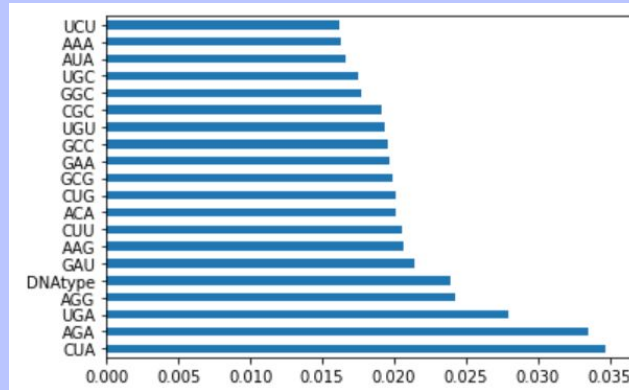




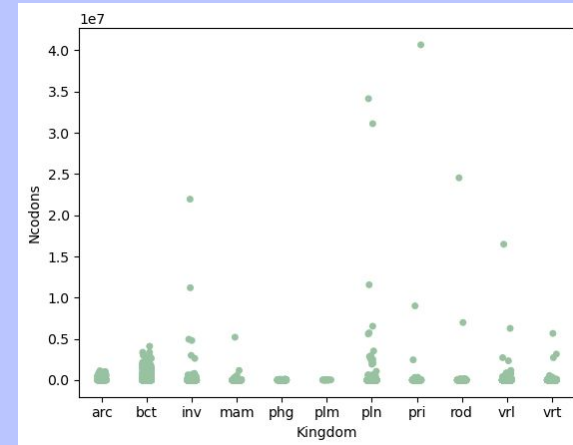
Distribution of the Kingdom values across the dataset



Distribution of the DNA Type values across the dataset



Feature Importance analysis to assess if DNA Type is a relevant feature



Outlier identification and relationship between codons and Kingdom

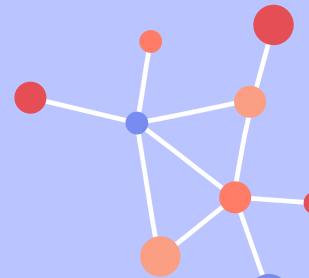


Data Cleaning & Preprocessing



This step is crucial, since untidy data will yield numerous errors and also inaccurate values from the analysis. Data exploration becomes difficult when the data is untidy. In our case we did the following:

- The non alphanumeric observations were removed first
- Non numeric values from codon columns were removed
- The data type of non float columns was changed
- Irrelevant columns such as 'SpeciesName' and 'SpeciesId' were removed





Feature Engineering

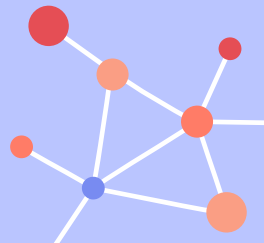
Normalization

ML can be more accurate when features have equal ranges.

Range of codons: 0 to 1

Range of 'NCodons': 1000 to 40,662,582

'NCodons' was normalized to the range 0 to 1 by dividing the single values by the maximum value.



Principal Component Analysis

To reduce number of factors, which could improve efficiency of ML algorithm

PCA tests data on inter-correlations and groups intercorrelated variables into components to reduce dimensions of data

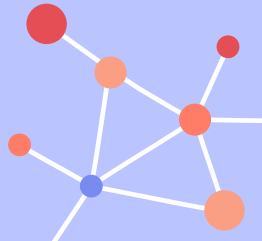
Testing adequacy of factoring:

Kaiser-Meyer-Olkin Test of sampling adequacy	Measure of sampling adequacy (MSA)	.14
Bartlett's Test of Homogeneity	Bartlett's K-Squared	234449
	df	63
	p	< .0001

Result:

- Homogenous variance
- $MSA < .6$

Not reducing dimensions based on low MSA



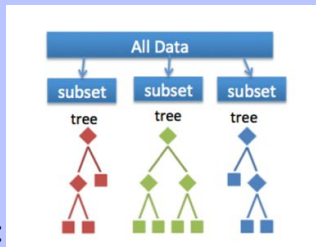
Modeling





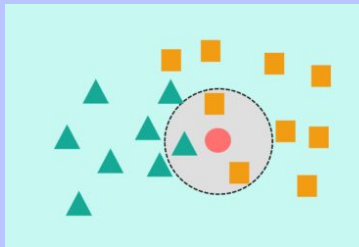
Random Forest

Great for classification problems, easy to use, ability to train model using different variables such as no. of trees, loss function and depth of tree. Ensemble algorithm: combines different algorithms. The different algorithms take a 'vote' to classify an object.



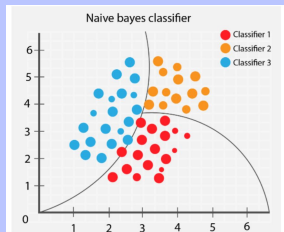
KNN

Simplest and widely used classification algorithm. Classification occurs by using the Euclidean distance of the new case from existing observations.



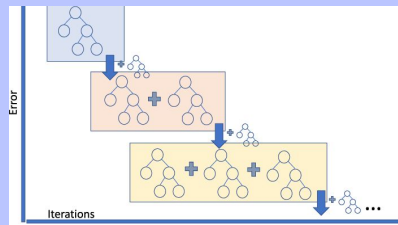
Naive Bayes

This algorithm assumes that the features are independent of each other. It usually performs well when the features are categorical



Gradient Boosting

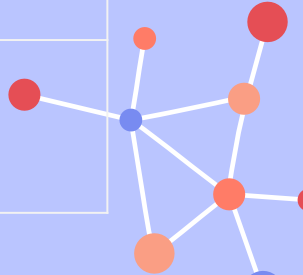
This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions.





Comparison of model performance

Model	Fit time (seconds)	Score time	Test score	Training score
Random Forest	8.79	0.07	.85	.94
K-nearest neighbor	.06	.46	.905	.938
Gradient Boost	169.4	.05	.881	.981
Linear SVC	1.506	0.0168	.8229	.832
Agglomerative Clustering	<<60	N/A == 39916800 permutations	?	?



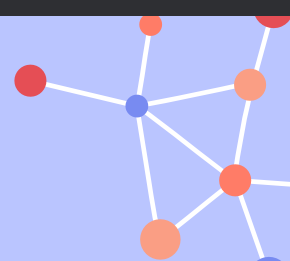


Hyperparameter tuning

For Hyperparameter tuning, the team changed various attribute values of the machine learning models. It was found that increasing the number of trees and depth of trees increased the accuracy. It was also found that the increase in `n_estimators` and decrease in `learning_rate` in a gradient boosting model increased the accuracy

```
n_trees = [5,10,20,50,100]
loss = ['gini', 'entropy', 'log_loss']
depth = [3,5,7,9]
for n in n_trees:
    for l in loss:
        for d in depth:
            steps = [ ('model', RandomForestClassifier(n_estimators = n, criterion = l, max_depth = d))]
            pipeline = Pipeline(steps=steps)
            # evaluate pipeline
            cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=2)
            scores = pd.DataFrame.from_dict( cross_validate(pipeline, df[feature_names], df['Kingdom'], cv=cv,
                                                            scoring='accuracy'),
                                             return_train_score=True))

            scores = scores.mean(axis=0)
            print ("When number of trees is ", n, " and loss function is ", l, " and the max depth is ", d)
            print(scores)
```

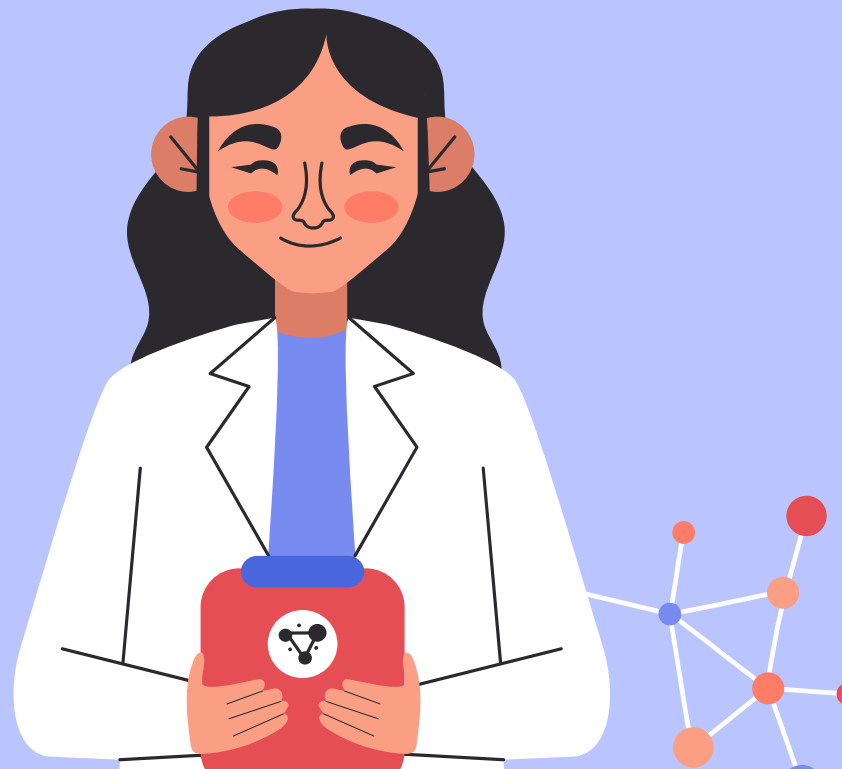




Conclusion

In conclusion, in our case the *Insert model name* performed the best out of all models trained in predicting the Kingdom given the features available.

Our model reached an accuracy of *insert*.





Citation

References:

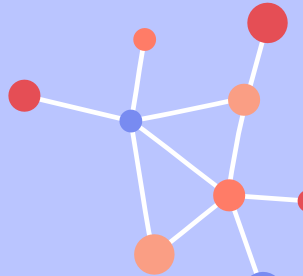
[1] <https://www.statisticshowto.com/kaiser-meyer-olkin/>

[2] Bartlett, M. S. (1937). "Properties of sufficiency and statistical tests". Proceedings of the Royal Statistical Society, Series A 160, 268–282 JSTOR 96803

[3] <https://builtin.com/data-science/random-forest-python-deep-dive>

[4] <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

[5] <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>





THANKS!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon** and infographics & images by **Freepik**

