Supervised machine learning: Identifying Ursidae among other families in Carnivora using COI sequences

Introduction

This paper is an extension of the Ursidae paper on the geographical distribution of the species and the species richness of the records found in the Barcode of life database (BOLD). This paper hopes to use supervised machine learning to build a classifier that can identify if a species is in the Ursidae family withing its higher classification, the order of carnivora. The first part of this bigger project used data from BOLD, whereas this paper will be analysing data from The National Center for Biotechnology Information database (NCBI). It investigates the mitochondrial gene, cytochrome c oxidase I (COI). The COI genes are of interest because of its proven viability in the global bio identification system of animals (Hebert et al., 2003).

The models of supervised learning implemented will be random forest and Generalized Boosted Regression Model (GBM). Both classification algorithms use decision trees (Glen, 2019). The difference is how the decision trees are built. Random forest builds trees independently, whereas, in gradient boosting the trees are built additively; one after another (Glen, 2019). Random forest will be implemented from the randomForest package and GBM will be implemented from the more robust caret package. Caret is an abbreviation for Classification And REgression Training (Prabhakaran, 2018). It is a multifunctional tool that can be used for preprocessing, visualization, training, tuning and predictions (Prabhakaran, 2018). It contains a plethora of machine learning methods. RandomForest can also be found in caret. Analysis included comparing the accuracy of both models and exploring the effect of the number of trees on accuracy.

Figures

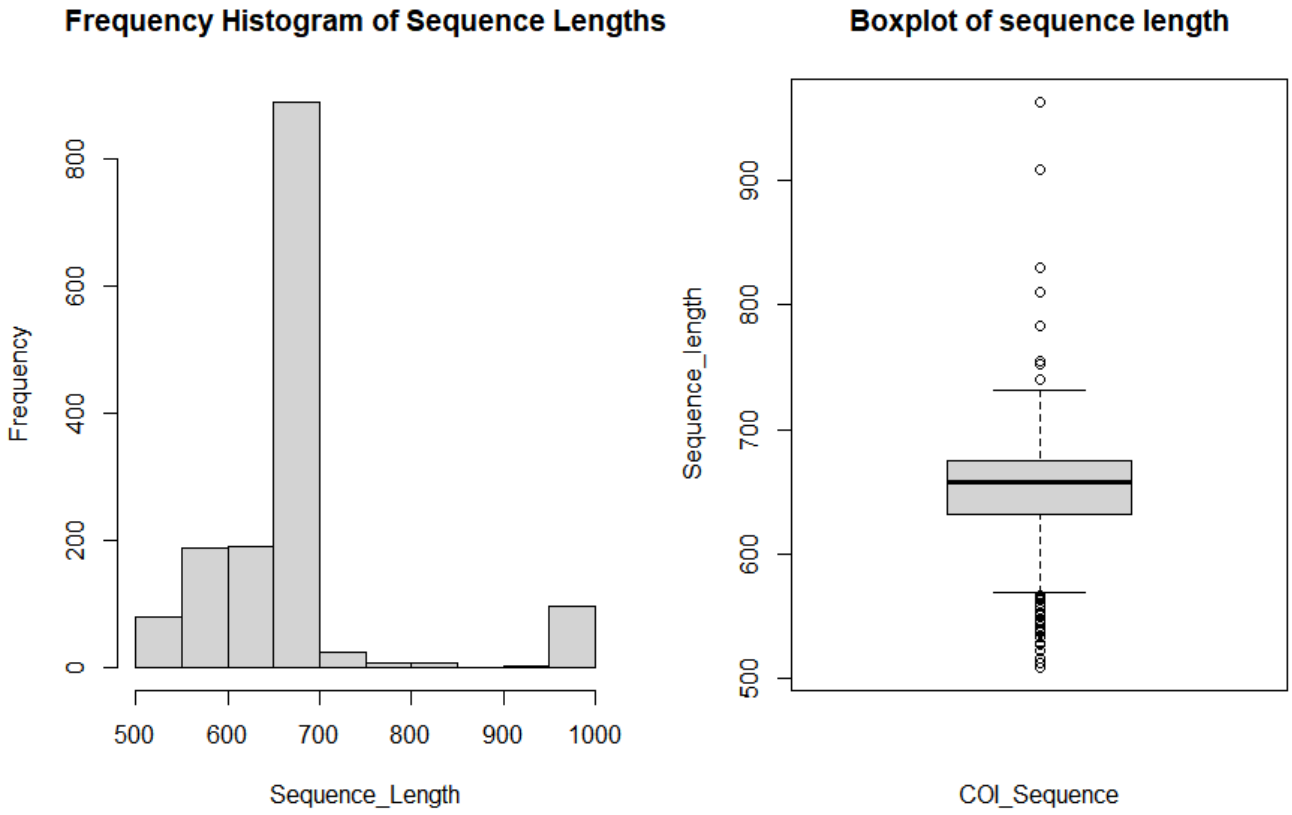

Figure 1. Histogram and box plot of the COI sequence lengths found in carnivora order after the sequence was cleaned but before sequence length was constrained.

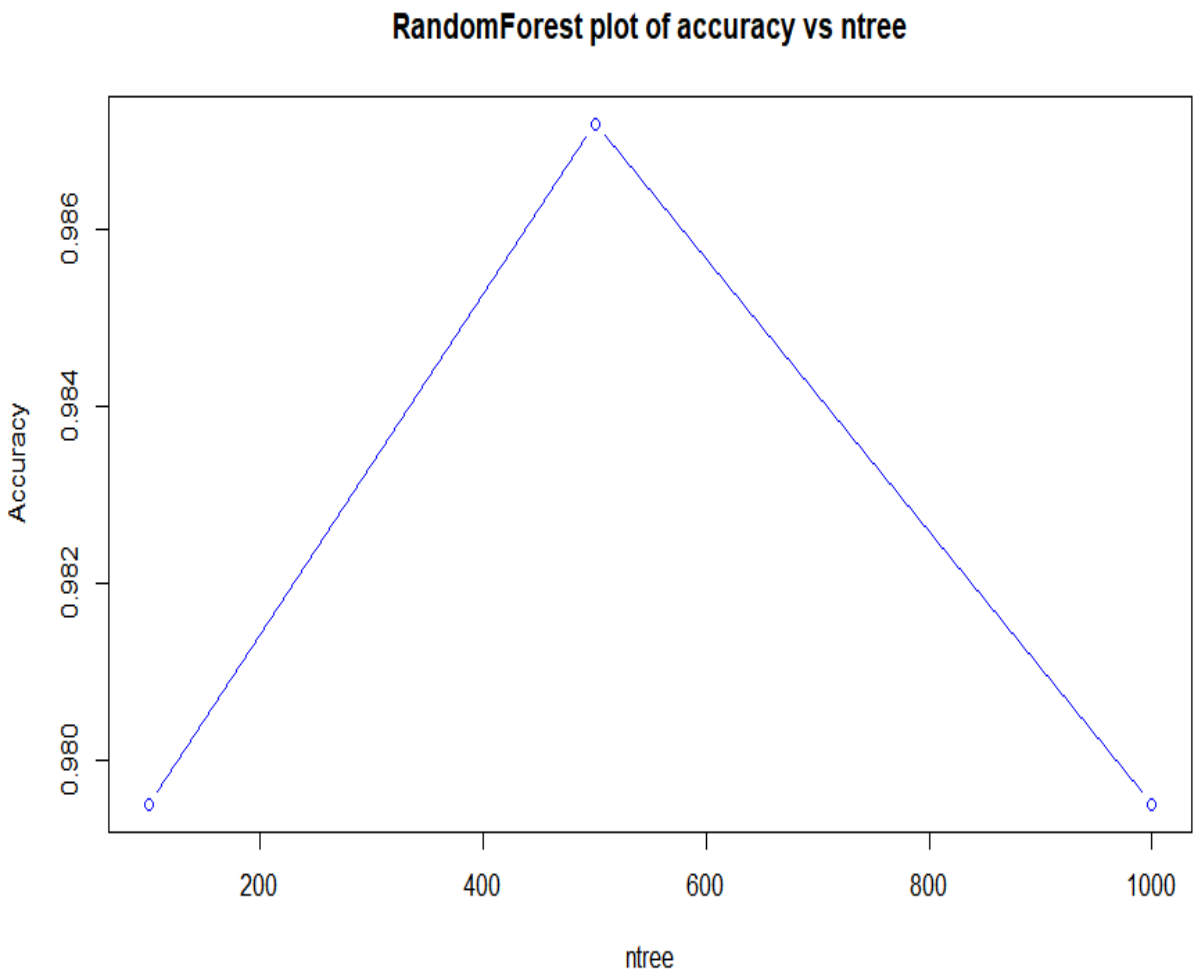

Figure 2. Line graph of the accuracy vs the number of trees used in randomForest

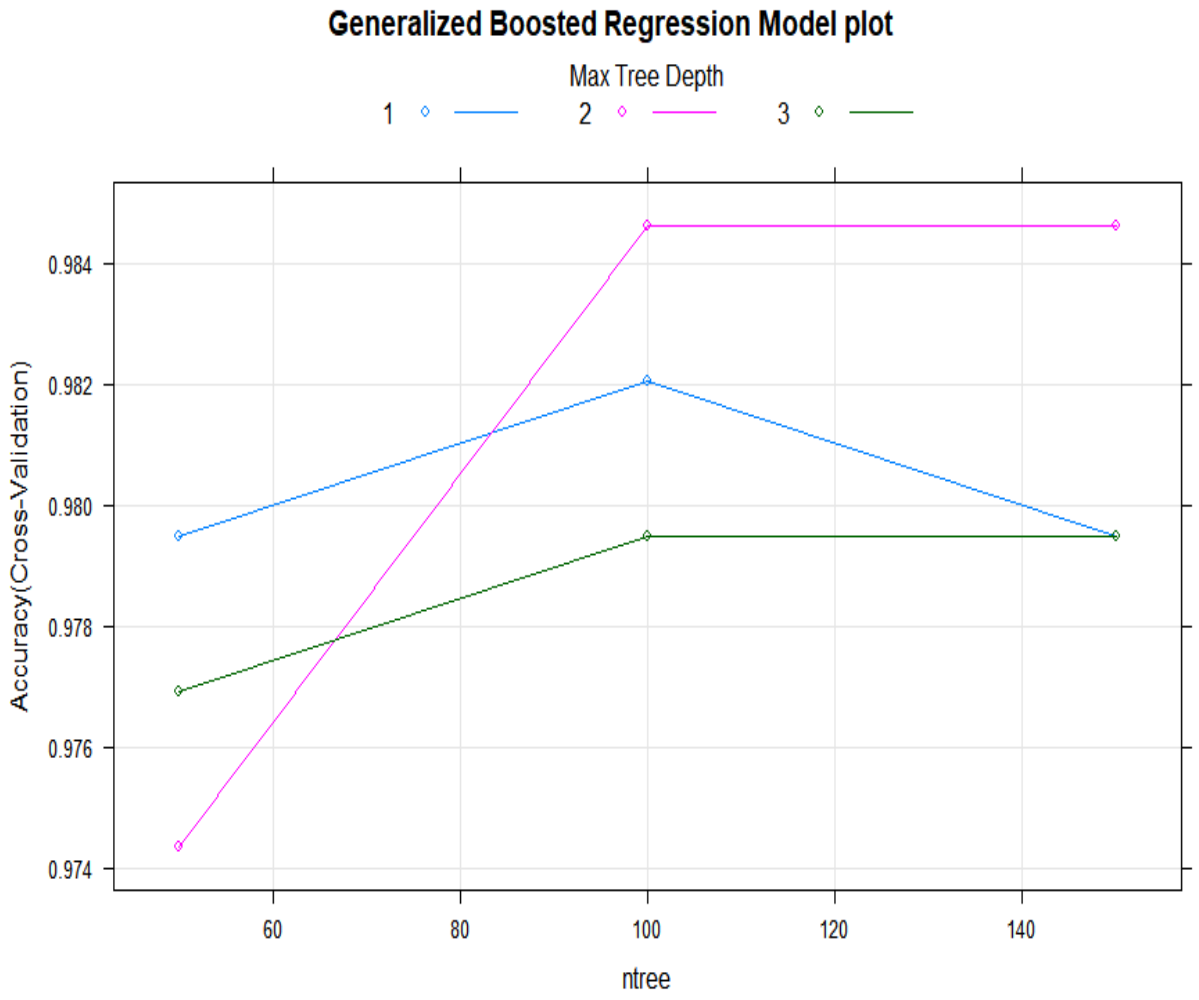

Figure 3. Line graph of the accuracy vs the number of trees used in the Generalized Boosted Regression model. Also showing relationship at different max tree depth.

Discussion and Conclusion

Initial preprosessing of the nucleotide data included removing any "-" or "N" that might have been at the front of the sequence or at the back. This was followed by removing all "-". This is done because we want the raw sequence information and not aligned sequences. We then omit any records where the percentage of remaining "N"s is greater than 1%. This 1% cut-off is adopted from the methods used Orton et al. (2019). Additionally, the sequence length variation was explored and reduced. As seen in **Figure 1.** , a histogram and boxplot of the sequence lengths was plotted. Subsequently, a subset of the records within the interquartile range was used for analysis. The proportions of A, T and G nucleotides, as well as the dinucleotide and trinucleotide frequencies were used to train the models.

Both the randomForest and Generalized boosted model performed well for the supervised machine learning. The randomForest had an accuracy of 98.21% while the GBM had an accuracy of 98.00%. The randomForest performed slightly better. The family of interest Ursidae, was predicted with a 100% accuracy using random forest. Gradient boosting can perform better than random forest but it is less appropriate for data with a lot of noise because it results in overfitting (Glen, 2019). The training and validation data were randomly sampled from the records. However, sampling with replacement was done because of under-representation of some families; Ailuridae had a sample size of 3, Odobenidae had 2, Otariidae had 4 and Procyonidae had 9, all out of 872 observations. This sampling with replacement is likely to produce noise in the data. Additionally, the paper probed at the relationship between the number of trees and the accuracy of the model. As seen in **Figure 2.** and **Figure 3.**, for both randomForest and gbm the accuracy of the models piques at an ntree value then drops or plateaus. The learning rate shouldn't be too big (small number of iterations) or too small (many iterations) (Bradley & Brandon, 2020).

In conclusion, the two models worked relatively well. Some limitations in the study arise from under sampling of the data. Aside from the underrepresentation of families found in the data, there is a lack of representation of some families and species. The carnivora order is comprised of 296 species and 16 families (Hassanin et al., 2021). This data set included 101 species and 13 families. Further sampling should improve accuracy of results. Future analysis might apply these models to different marker genes and compare the accuracy. It would also be interesting to use unsupervised machine learning in the Ursidae family and compare the number of clusters to the number of BINs (Barcode Index Numbers) and species in the BOLD database.

Acknowledgments

I would like to acknowledge the following people for their contributions to this project:

Jessica Labarcena – for your endless patience with dealing with our problems

Gibran Edun – for your assistance in resolving a problem with tax_name

Nishita Sharif – for bringing to my attention that it was possible to combine plots in R

I would also like to acknowledge Daniel Amoako, Amjad Osman, Jesse, and Omar Khan and Kazra for engaging in discussion pertaining to this project.

References

A Gentle Introduction to k-fold Cross-Validation (2018). Machinelearningmastery. Retrieved October 27, 2022, from https://machinelearningmastery.com/k-fold-cross-validation/

Bradley, B. & Brandon G. (2020). *Hands-On Machine Learning with R*. Github. https://bradleyboehmke.github.io/HOML/

Exploring the caret package. (n.d.). Rstudio. Retrieved October 27, 2022, from http://rstudio-pubs-static.s3.amazonaws.com/251240_12a8ecea8e144fada41120ddcf52b116.html

Glen, S. (2019). *Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply.* Datasciencecentral. https://www.datasciencecentral.com/decision-tree-vs-random-forest-vs-boosted-trees-explained/

Hassanin, A., Veron, G., Ropiquet, A., van Vuuren, B. J., Lecu, A., Goodman, S. M., Haider, J., & Trung Thanh Nguyen. (2021). Evolutionary history of Carnivora (Mammalia, Laurasiatheria) inferred from mitochondrial genomes. *PloS One*, *16*(2), e0240770–e0240770. https://doi.org/10.1371/journal.pone.0240770

Hebert, P. D. N., Cywinska, A., Ball, S. L., & deWaard, J. R. (2003). Biological identifications through DNA barcodes. *Proceedings of the Royal Society. B, Biological Sciences*, *270*(1512), 313–321. https://doi.org/10.1098/rspb.2002.2218

Orton, M. G., May, J. A., Ly, W., Lee, D. J., & Adamowicz, S. J. (2019). Is molecular evolution faster in the tropics? *Heredity*, *122*(5), 513–524. https://doi.org/10.1038/s41437-018-0141-7

Prabhakaran, S. (2018). *Caret Package – A Practical Guide to Machine Learning in R*. Machinelearningplus. https://www.machinelearningplus.com/machine-learning/caret-package/

R.Script

```
####Introduction####
#This paper is an extension of the Ursidae paper on the geographical
distribution of the species and the species richness of the records found in
the Barcode of life database (BOLD). This paper hopes to use supervised
machine learning to build a classifier that can identify if a species is in
the Ursidae family withing its higher classification, the order of carnivora.
The first part of this bigger project used data from BOLD, whereas this paper
will be analysing data from The National Center for Biotechnology Information
database (NCBI). It investigates the mitochondrial gene, cytochrome c oxidase
I (COI). The COI genes are of interest because of its proven viability in the
global bio identification system of animals (Hebert et al., 2003).
#The models of supervised learning implemented will be random forest and
Generalized Boosted Regression Model (GBM). Both classification algorithms use
decision trees (Glen, 2019). The difference is how the decision trees are
built. Random forest builds trees independently, whereas, in gradient boosting
the trees are built additively; one after another (Glen, 2019). Random forest
will be implemented from the randomForest package and GBM will be implemented
from the more robust caret package. Caret is an abbreviation for
Classification And REgression Training (Prabhakaran, 2018). It is a
multifunctional tool that can be used for preprocessing, visualization,
training, tuning and predictions (Prabhakaran, 2018). It contains a plethora
of machine learning methods. RandomForest can also be found in caret. Analysis
included comparing the accuracy of both models and exploring the effect of the
number of trees on accuracy.


####setting up####
###install packages
# install.packages("taxize")
# install.packages("caret")

###load packages
library(tidyverse)
library(Biostrings)
library(rentrez)
library(taxize)
library(sys)
library(caret)

###Get present Working Directory. Previously set working directory with
setwd(). Excluded runnable function to prevent error.
getwd()




####Data exploration####

###search and fetch needed information from nuccore database
##database search of nuccore. Organism is carnivora, gene is (COI or COX1) and
sequence length is between 500-1000. Web history rather than retmax because of
the large amounts of sequences from NCBI. Used by setting "use_history"
argument to TRUE.
```

```
dbsearch_nuccore_Carnivora <- entrez_search(db = "nuccore", term =
"((carnivora[Organism]) AND 500:1000[Sequence Length]) AND (COI OR COX1)",
use_history = T)
dbsearch_nuccore_Carnivora
#get number of returned searches
length(dbsearch_nuccore_Carnivora$ids)
#get count of all search results
dbsearch_nuccore_Carnivora$count
#fetch fasta files of the database search using web_history to fetch
fetch_carnivora <- entrez_fetch(db = "nuccore", web_history =
dbsearch_nuccore_Carnivora$web_history, rettype = "fasta")
class(fetch_carnivora)
fetch_carnivora
#write fasta to hard drive and separate fields by \n (comes before identifier)
write(fetch_carnivora, "carnivora_fetch.fasta", sep = "\n")
```

```
###Make dataframe with needed information
#read fasta file back as DNA StringSet
DNAStringset_carnivora <- readDNAStringSet("carnivora_fetch.fasta")
class(DNAStringset_carnivora)
head(names(DNAStringset_carnivora))
#convert information from DNAStringset to dataframe for further manipulation.
Naming the column for the header as header_identifier and the columns for the
sequences as COI_sequence
df_carnivora_seq <- data.frame(header_identifier =
names(DNAStringset_carnivora), COI_sequence = paste(DNAStringset_carnivora))
View(df_carnivora_seq)
#make a new column called species name. The species name are the second and
third terms in the header identifier
df_carnivora_seq$Species_Name <- word(df_carnivora_seq$header_identifier, 2L,
3L)
#get number of species
length(unique(df_carnivora_seq$Species_Name))
#create new column for genus names
df_carnivora_seq$Genus_name <- word(df_carnivora_seq$Species_Name, 1L)
#get number of genus
length(unique(df_carnivora_seq$Genus_name))


# ###The following has some novel functions that run into errors obtaining
information through an API. Data obtained from code is read into R below.
# ###Create a column for Family name. Family is ultimately what we are trying
to classify. The family name is not included in the fasta file.Here we use a
function called tax_name from taxize package to get the family names from the
species name. After each species family is identified, we will assign the
family names accordingly.
```

```
# #Get the families for the unique species. Running tax_name on the smaller
record of the unique sequences rather than all species because tax_name is
more likely to run into error working with API for longer records.
# unique_species <- unique(df_carnivora_seq$Species_Name)
# family_for_unique_species <- tax_name(unique_species, get = "family", db =
"ncbi")
# #make a vector with the family assignment of each species (for each record
not unqiue species) in the carnivora dataframe. Name vector
family_assignments_for_df_carnivora
# family_assignments_for_df_carnivora <- c()
# for (i in df_carnivora_seq$Species_Name){
#   #temp is a temporary dataframe to hold family name
#   temp <- family_for_unique_species%>%
#     #query is species name searched against database (check
family_for_unique_species or ?tax_name)
#     filter(query==i)
#   #add the family name for that species to
family_assignments_for_df_carnivora vector
#   family_assignments_for_df_carnivora <-
append(family_assignments_for_df_carnivora, temp$family)
# }
# #remove objects made because of loop
# rm(i, temp)
# # make family_assignments_for_df_carnivora vector as the column for the
family name
# df_carnivora_seq$Family_name <- family_assignments_for_df_carnivora
# View(df_carnivora_seq)
# #re-arrange columns
# names(df_carnivora_seq)
# df_carnivora_seq <- df_carnivora_seq[ ,c("header_identifier", "Family_name",
"Genus_name", "Species_Name", "COI_sequence")]
# #write_csv(df_carnivora_seq,"NCBI_needed_info.csv")
df_carnivora_seq <- read.csv("NCBI_needed_info.csv")
#how many family_names are in our data
length(unique(df_carnivora_seq$Family_name))
#how many species are in our data
length(unique(df_carnivora_seq$Species_Name))



###check for outliers
#outliers in family column. The family column should not contain any value
that is not Ailuridae, Canidae, Eupleridae, Felidae, Herpestidae, Hyaenidae,
Mephitidae, Mustelidae, Nandiniidae, Odobenidae, Otariidae, Phocidae,
Procyonidae, Ursidae, Viverridae.
df_carnivora_seq %>%
  filter(!Family_name %in% c("Ailuridae", "Canidae", "Eupleridae", "Felidae",
"Herpestidae", "Hyaenidae", "Mephitidae", "Mustelidae", "Nandiniidae",
"Odobenidae", "Otariidae", "Phocidae", "Procyonidae", "Ursidae",
"Viverridae")) %>%
  print()
#outliers in sequence length
boxplot(nchar(df_carnivora_seq$COI_sequence), xlab = "COI_Sequence", ylab =
"Sequence_length", main = "Boxplot of COI sequence length")
```

```
#check for any Na's
sum(is.na(df_carnivora_seq))




###clean and filter nucleotide data.
##Create a new nucleotide column called COI_sequence2.
df_carnivora_seq <- df_carnivora_seq %>%
  #create new column while keeping the old column and remove leading "-" or
"N"s
  mutate(COI_sequence2 = str_remove(COI_sequence, "^[-N]+")) %>%
  #remove trailing "-" or "N"s
  mutate(COI_sequence2 = str_remove(COI_sequence2, "[-N]+$")) %>%
  #remove all "-"s
  mutate(COI_sequence2 = str_remove_all(COI_sequence2, "-+")) %>%
  #remove records that have "N"s that make up more than 1% of original
sequence
  filter(str_count(COI_sequence2, "N") <= (0.01 * str_count(COI_sequence)))
View(df_carnivora_seq)
#take a look at the distribution of the sequence lengths. Create a dataframe
copy of the sequence lengths before filtering for sequence lengths for
analysis later later.
df_carnivora_seq_beforesubset <- df_carnivora_seq
hist(nchar(df_carnivora_seq_beforesubset$COI_sequence), xlab =
"Sequence_Length", ylab = "Frequency", main = "Frequency Histogram of COI
Sequence Lengths")
summary(nchar(df_carnivora_seq$COI_sequence))
#Assign a vector to hold the first quartilie and third quartile of sequence
lengths
q1 <- quantile(nchar(df_carnivora_seq$COI_sequence2), probs = 0.25, na.rm =
TRUE)
q1

q3 <- quantile(nchar(df_carnivora_seq$COI_sequence2), probs = 0.75, na.rm =
TRUE)
q3
#Filter records to only include COI sequences that are inbetween the
interquartile range.
df_carnivora_seq <- df_carnivora_seq %>%
  filter(str_count(COI_sequence2) >= q1 & str_count(COI_sequence2) <= q3)
#Checks to make sure everything worked as expected
summary(str_count(df_carnivora_seq$COI_sequence2))




###Calculate sequence features
#convert COI sequence to DNAStringset so that we can use Biostrings package.
```

```
df_carnivora_seq$COI_sequence2 <- DNAStringSet(df_carnivora_seq$COI_sequence2)
class(df_carnivora_seq$COI_sequence2)
class(df_carnivora_seq)
##calculate nucleotide frequencies.
#Add column of the absolute count of A, C , G, and T in the COI_sequence2
column using letterFrequency. Use cbind to append to df_carnivora_seq
df_carnivora_seq <- cbind(df_carnivora_seq,
as.data.frame(letterFrequency(df_carnivora_seq$COI_sequence2, letters = c("A",
"C","G", "T"))))
View(df_carnivora_seq)
#Add the proportional frequencies of the nucleotides in proportion to the
other nucleotides. Creating new columns using "$"
df_carnivora_seq$Aprop <- (df_carnivora_seq$A) / (df_carnivora_seq$A +
df_carnivora_seq$C + df_carnivora_seq$G + df_carnivora_seq$T)

df_carnivora_seq$Tprop <- (df_carnivora_seq$T) / (df_carnivora_seq$A +
df_carnivora_seq$C + df_carnivora_seq$G + df_carnivora_seq$T)

df_carnivora_seq$Gprop <- (df_carnivora_seq$G) / (df_carnivora_seq$A +
df_carnivora_seq$C + df_carnivora_seq$G + df_carnivora_seq$T)

View(df_carnivora_seq)
#Add dinucleotide and trinucleotide frequencies
df_carnivora_seq <- cbind(df_carnivora_seq,
as.data.frame(dinucleotideFrequency(df_carnivora_seq$COI_sequence2, as.prob =
TRUE)))

df_carnivora_seq <- cbind(df_carnivora_seq,
as.data.frame(trinucleotideFrequency(df_carnivora_seq$COI_sequence2, as.prob =
TRUE)))
#check to see everything added
names(df_carnivora_seq)
View(df_carnivora_seq)
```

```
####Analysis to address questions####
###Using randomforest to identify Families in the carnivora order
###Training Random forest classification model
#Change COI_sequence2 to character from Biostring
df_carnivora_seq$COI_sequence2 <- as.character(df_carnivora_seq$COI_sequence2)
class(df_carnivora_seq$COI_sequence2)
#check counts by family name
table(df_carnivora_seq$Family_name)
##Get a randomized sample of records. Randomize by the family name. Set seed
so results are reproducible.
```

```
#Get records for validating the classifier. Records sampled with replacement
because some families are severely underrepresented with small sample sizes of
2, 3, 4 and 9. Total data for each family will be 40. 25% for validation (10)
and 75% for training (30)
set.seed(999)
dfValidation_COI <- df_carnivora_seq %>%
  group_by(Family_name) %>%
  sample_n(10, replace = T)
#save value of seed in a vector for future use
seed_for_dfValidation_COI <- 999
#check if evenly split and each family represented
table(dfValidation_COI$Family_name)
#Get records for training the classifier.
set.seed(888)
dfTraining_COI <- df_carnivora_seq %>%
  group_by(Family_name) %>%
  sample_n(30, replace = T)
#save value of seed in a vector for future use
seed_for_dfTraining_COI <- 888
#check if evenly split and each family represented
table(dfTraining_COI$Family_name)
#Build classifier to separate Family_names. Using A, T, and G proportions as
well as dinucleotide and tricnucleotide frequencies as predictors. The
response variable is Family_name.
names(df_carnivora_seq)

COI_randomforest_classifier <- randomForest::randomForest(x = dfTraining_COI[,
11:93], y = as.factor(dfTraining_COI$Family_name), ntree = 100, importance =
TRUE)

#View some specifics of the random forest classifier
COI_randomforest_classifier
COI_randomforest_classifier$importance
COI_randomforest_classifier$err.rate
#View confusion matrix
View(COI_randomforest_classifier$confusion)

##get accuracy of model. accuracy = 1 - error rate
#store value of OBB error rate
COI_randomforest_classifier
OBB_error_rate <- 0.0179
#store accuracy in vector for later anlysis
randomForest_accuracy <- 1 - OBB_error_rate

###Testing classification model
#Test classifier with Validation dataframe
COI_predict_Validation <- predict(COI_randomforest_classifier,
dfValidation_COI[ , c(2, 11:93)])
##Check result of prediction
#check properties of prediction
COI_predict_Validation
class(COI_predict_Validation)
length(COI_predict_Validation)
```

```
#create confusion matrix of prediction. Setting columns as observed and rows
as predicted.
table(observed = dfValidation_COI$Family_name, predicted =
COI_predict_Validation)
```

```
###How does a different model compare to the random forest?
###Training a Generalized Boosted Regression Models (GBM) with the more robust
Caret package
#subset data to only include predictor and response variables
GBM_dfValidation <- dfValidation_COI[ , c(2, 11:93)]
GBM_dfTraining <- dfTraining_COI[ , c(2, 11:93)]
class(GBM_dfValidation)
class(GBM_dfValidation)
#change class to data frame
GBM_dfValidation <- as.data.frame(GBM_dfValidation)
GBM_dfTraining <- as.data.frame(GBM_dfTraining)
class(GBM_dfValidation)
class(GBM_dfValidation)
#Define train control. the method will be cross validation. the number of
folds(cuts) will be set to 10.
myControl <- trainControl(
              method = "cv",
              number = 10,
              classProbs = TRUE,
              #simplify for readability
              verboseIter = FALSE)
#train model. object on the left of "~." is predictor variable column
name.Object to the right is a dataframe including both predictors and response
variables
set.seed(222)
GBM_model <- train(Family_name ~., data = GBM_dfTraining,
               method = "gbm",
               metric ="Accuracy",
               trControl = myControl)
#save value of seed in a vector for future use
seed_for_GBMTraining_COI <- 222
## Print model to console
GBM_model
#calculate overall accuracy of GBM model
GBM_accuracy <- mean(GBM_model$results$Accuracy)
#Validating with predict
set.seed(111)
GBM_predict_validation <- predict(GBM_model, GBM_dfValidation, type = "raw")
#save value of seed in a vector for future use
seed_for_GBMTraining_COI <- 111
```

```
#create confusion matrix of prediction. Setting columns as observed and rows
as predicted.
table(observed = GBM_dfValidation$Family_name, predicted =
GBM_predict_validation)
```

```
###How does accuracy change with number of iterations?
##perform random forest 3 times with different ntree. use ntree (100, 500,
1000)
data1_randomforest_plot <- randomForest::randomForest(x = dfTraining_COI[,
11:93], y = as.factor(dfTraining_COI$Family_name), ntree = 100, importance =
TRUE)

data2_randomforest_plot <- randomForest::randomForest(x = dfTraining_COI[,
11:93], y = as.factor(dfTraining_COI$Family_name), ntree = 500, importance =
TRUE)

data3_randomforest_plot <- randomForest::randomForest(x = dfTraining_COI[,
11:93], y = as.factor(dfTraining_COI$Family_name), ntree = 1000, importance =
TRUE)

###store ntree and accuracy numbers
##get accuracy of models. accuracy = 1 - error rate
#store value of OBB error rate
data1_randomforest_plot
data1_error_rate <- 0.0205

data2_randomforest_plot
data2_error_rate <- 0.0128

data3_randomforest_plot
data3_error_rate <- 0.0205
#store accuracy values
data1_accuracy <- 1 - data1_error_rate
data2_accuracy <- 1 - data2_error_rate
data3_accuracy <- 1 - data3_error_rate
#store accuracy and ntree number in one data frame
randomForest_ntree_vs_accuracy <- data.frame(c(100, 500, 1000),
c(data1_accuracy, data2_accuracy, data3_accuracy), stringsAsFactors = TRUE)
#assign names to variables
names(randomForest_ntree_vs_accuracy) <- c("ntree", "Accuracy")
View(randomForest_ntree_vs_accuracy)
#plot ntree vs accuracy with accuracy on y axis and ntree on x axis.
plot(randomForest_ntree_vs_accuracy, xlab = "ntree", ylab = "Accuracy", main =
"RandomForest plot of accuracy vs ntree", type = "b", col = "blue")
```

```
##Generalized Boosted Regression Model (GBM) plot
plot(GBM_model, xlab = "ntree", ylab = "Accuracy(Cross-Validation)", main =
"Generalized Boosted Regression Model plot")
```

```
####plots to submit####

###RadomForest ntree vs accuracy plot
plot(randomForest_ntree_vs_accuracy, xlab = "ntree", ylab = "Accuracy", main =
"RandomForest plot of accuracy vs ntree", type = "b", col = "blue")


###GBM plot
plot(GBM_model, xlab = "ntree", ylab = "Accuracy(Cross-Validation)", main =
"Generalized Boosted Regression Model plot")
```

```
#combine two plots into one using par(). Allows you to specify rows and
columns. will be using one row and two columns
```

```
par(mfrow = c(1,2))

##Histogram of sequence length
plot1 <- hist(nchar(df_carnivora_seq_beforesubset$COI_sequence), xlab =
"Sequence_Length", ylab = "Frequency", main = "Frequency Histogram of Sequence
Lengths")

##Barplot of sequence length showing q1 and q3
plot2 <- boxplot(nchar(df_carnivora_seq_beforesubset$COI_sequence), xlab =
"COI_Sequence", ylab = "Sequence_length", main = "Boxplot of sequence length")
```

####Results and discussion####
#Initial preprosessing of the nucleotide data included removing any "-" or "N" that might have been at the front of the sequence or at the back. This was followed by removing all "-". This is done because we want the raw sequence information and not aligned sequences. We then omit any records where the percentage of remaining "N"s is greater than 1%. This 1% cut-off is adopted from the methods used Orton et al. (2019). Additionally, the sequence length variation was explored and reduced. As seen in Figure 1. , a histogram and boxplot of the sequence lengths was plotted. Subsequently, a subset of the records within the interquartile range was used for analysis. The proportions of A, T and G nucleotides, as well as the dinucleotide and trinucleotide frequencies were used to train the models.
#Both the randomForest and Generalized boosted model performed well for the supervised machine learning. The randomForest had an accuracy of 98.21% while the GBM had an accuracy of 98.00%. The randomForest performed slightly better. The family of interest Ursidae, was predicted with a 100% accuracy using random forest. Gradient boosting can perform better than random forest but it is less appropriate for data with a lot of noise because it results in overfitting (Glen, 2019). The training and validation data were randomly sampled from the records. However, sampling with replacement was done because of under-representation of some families; Ailuridae had a sample size of 3, Odobenidae had 2, Otariidae had 4 and Procyonidae had 9, all out of 872 observations. This sampling with replacement is likely to produce noise in the data. Additionally, the paper probed at the relationship between the number of trees and the accuracy of the model. As seen in Figure 2. and Figure 3., for both randomForest and gbm the accuracy of the models piques at an ntree value then drops or plateaus. The learning rate shouldn't be too big (small number of iterations) or too small (many iterations) (Bradley & Brandon, 2020).

#In conclusion, the two models worked relatively well. Some limitations in the study arise from under sampling of the data. Aside from the underrepresentation of families found in the data, there is a lack of representation of some families and species. The carnivora order is comprised of 296 species and 16 families (Hassanin et al., 2021). This data set included 101 species and 13 families. Further sampling should improve accuracy of results. Future analysis might apply these models to different marker genes and compare the accuracy. It would also be interesting to use unsupervised machine learning in the Ursidae family and compare the number of clusters to the number of BINs (Barcode Index Numbers) and species in the BOLD database.

####Acknowledgments####
#I would like to acknowledge the following people for their contributions to this project:
Jessica Labarcena - for your endless patience with dealing with our problems
#Gibran Edun - for your assistance in resolving a problem with tax_name
#Nishita Sharif - for bringing to my attention that it was possible to combine plots in R
#I would also like to acknowledge Daniel Amoako, Amjad Osman, Jesse, and Omar Khan and Kazra for engaging in discussion pertaining to this project.

#Initial preprosessing of the nucleotide data included removing any "-" or "N" that might have been at the front of the sequence or at the back. This was followed by removing all "-". This is done because we want the raw sequence information and not aligned sequences. We then omit any records where the percentage of remaining "N"s is greater than 1%. This 1% cut-off is adopted from the methods used Orton et al. (2019). Additionally, the sequence length variation was explored and reduced. As seen in Figure 1. , a histogram and boxplot of the sequence lengths was plotted. Subsequently, a subset of the records within the interquartile range was used for analysis. The proportions of A, T and G nucleotides, as well as the dinucleotide and trinucleotide frequencies were used to train the models.
#Both the randomForest and Generalized boosted model performed well for the supervised machine learning. The randomForest had an accuracy of 98.21% while the GBM had an accuracy of 98.00%. The randomForest performed slightly better. The family of interest Ursidae, was predicted with a 100% accuracy using random forest. Gradient boosting can perform better than random forest but it is less appropriate for data with a lot of noise because it results in overfitting (Glen, 2019). The training and validation data were randomly sampled from the records. However, sampling with replacement was done because of under-representation of some families; Ailuridae had a sample size of 3, Odobenidae had 2, Otariidae had 4 and Procyonidae had 9, all out of 872 observations. This sampling with replacement is likely to produce noise in the data. Additionally, the paper probed at the relationship between the number of trees and the accuracy of the model. As seen in Figure 2. and Figure 3., for both randomForest and gbm the accuracy of the models piques at an ntree value then drops or plateaus. The learning rate shouldn't be too big (small number of iterations) or too small (many iterations) (Bradley & Brandon, 2020).

#In conclusion, the two models worked relatively well. Some limitations in the study arise from under sampling of the data. Aside from the underrepresentation of families found in the data, there is a lack of representation of some families and species. The carnivora order is comprised of 296 species and 16 families (Hassanin et al., 2021). This data set included 101 species and 13 families. Further sampling should improve accuracy of results. Future analysis might apply these models to different marker genes and compare the accuracy. It would also be interesting to use unsupervised machine learning in the Ursidae family and compare the number of clusters to the number of BINs (Barcode Index Numbers) and species in the BOLD database.

####References####

#A Gentle Introduction to k-fold Cross-Validation (2018). Machinelearningmastery. Retrieved October 27, 2022, from https://machinelearningmastery.com/k-fold-cross-validation/

Bradley, B. & Brandon G. (2020). Hands-On Machine Learning with R. Github. https://bradleyboehmke.github.io/HOML/

Exploring the caret package. (n.d.). Rstudio. Retrieved October 27, 2022, from http://rstudio-pubs-static.s3.amazonaws.com/251240_12a8ecea8e144fada41120ddcf52b116.html

#Glen, S. (2019). Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply. Datasciencecentral. https://www.datasciencecentral.com/decision-tree-vs-random-forest-vs-boosted-trees-explained/

Hassanin, A., Veron, G., Ropiquet, A., van Vuuren, B. J., Lecu, A., Goodman, S. M., Haider, J., & Trung Thanh Nguyen. (2021). Evolutionary history of Carnivora (Mammalia, Laurasiatheria) inferred from mitochondrial genomes. PloS One, 16(2), e0240770–e0240770. https://doi.org/10.1371/journal.pone.0240770

#Hebert, P. D. N., Cywinska, A., Ball, S. L., & deWaard, J. R. (2003). Biological identifications through DNA barcodes. Proceedings of the Royal Society. B, Biological Sciences, 270(1512), 313–321. https://doi.org/10.1098/rspb.2002.2218

#Orton, M. G., May, J. A., Ly, W., Lee, D. J., & Adamowicz, S. J. (2019). Is molecular evolution faster in the tropics? Heredity, 122(5), 513–524. https://doi.org/10.1038/s41437-018-0141-7

#Prabhakaran, S. (2018). Caret Package – A Practical Guide to Machine Learning in R. Machinelearningplus. https://www.machinelearningplus.com/machine-learning/caret-package/