STAT GU4241/GR5241
Professor: Banu Baydil
Group Project
Spring 2023

Application of Statistical Machine Learning Methods on the Analysis of Predator and Prey Dataset

Group Name:
Marine Species Researchers

Group Members:
Jacob Minkin (jcm2284)
Jingyi Zhang (jz3543)
Nixon Mckenzie (nnm2132)
Xinyue Yuan (xy2555)
Yajie Lou (yl4386)

Abstract

Analyzing predators and prey can provide valuable insights into ecological relationships and dynamics within an ecosystem. These analyses can help us to identify which species may be at risk of decline or extinction due to changes in predator or prey populations, habitat loss, climate change, or other factors.

In the classification modeling part, we predict the common name of predators and the taxon of prey. By applying SVC, Decision Tree, Random Forest, and Adaboost models, they were trained on the training dataset, and their performance was evaluated by calculating the train and test scores. Grid search with three-fold cross-validation was used to tune the hyperparameters of the models. The feature importance of the best models was analyzed, and bar charts were plotted to visualize the results. The significant features for predicting the common name of predators are specific habitat, geographic location, mean primary productivity, mean annual temperature, predator length, and predator mass. For predicting the taxon of prey, the important features are prey length, prey mass, and specific habitat.

In order to focus on the most significant features, the PCA method was applied to reduce the dataset to 3D. The reduced dataset was visualized using a 3D scatter plot and the projection of the scatter points on the three dimensions. Unfortunately, the PCA transformed data did not provide a better result, so it was not used in the subsequent machine learning process.

In the clustering part, the unsupervised clustering method K-means was applied to evaluate the predator and prey datasets. The number of groups was determined based on the elbow method of the inertia graph. The bluefin tuna was separated as a sub-group in the predator dataset, while prey with a mixed taxon was easily separated due to differences in living depth.

In the regression modeling part, we investigate the relationship between the length and mass of prey and the biological and geological features of their predators. By utilizing Linear, Ridge, Lasso, Decision Tree, AdaBoost, and Random Forest Regression. The resulting data was split into training and testing sets, and then scaled using a standard scalar. The significant features for predicting prey mass are Latitude Minute and Geographic Location. For predicting the prey length, the important feature is Latitude minute.

The classification, clustering, and regression analysis of predator and prey dataset has an important real life meaning, which helps people to better understand the interactions between predators and prey can help us to better understand the food webs, population dynamics, and biodiversity of an ecosystem. We will continually improve our analysis by considering a more comprehensive dataset and applying some more advanced statistical methods, hoping to bring our insights into the field of biology and make positive influences on the interaction between predator and prey!

Ⅰ. Problem and Motivation

The goal of this project has three parts: Classification, Clustering, and Regression. For the classification part, the main goal is to predict the predator's common name and prey's taxon based on biological and geological features. Then evaluate the performance of the model and compare the predictions with the true values. For the clustering part, the unsupervised clustering method K-means was applied to evaluate the predator and prey datasets. For the regression part, the main purpose is to explore whether the length and mass of various prey is related to predators' biological and geological characteristics. If they are related, then how and to what extent predators' characteristics affect their length. To achieve this, two separate linear regression models were built, with the first response variable being prey mass (Y1) and the second response variable being prey length (Y2). The predictors (X) used in both models were the predators' general biological and geological features.

Ⅱ. Data

Our data set is from the websitehttps://retriever.readthedocs.io/en/latest/datasets_list.html, dataset #94: Marine Predator and Prey Body Sizes - Barnes et al. 2008

- Dataset has 61 columns * 34,932 records
- Dataset is collected within the time period: 2006 ~ 2014. The variables are listed below.

Table 1: The Description of Variables

Variable	Description
Record Number	Record Number
Individual ID	Individual predator ID
Predator	Genus species of predator
Predator common name	Common name of predator
Predator taxon	Ectotherm vertebrate or cephalopod
Predator lifestage	Larva, postlarva, juvenile or adult
Type of feeding interaction	Insectivorous, piscivorous, planktivorous, or predacious
Predator length	Predator length
Predator length unit	μm, mm or cm
Predator standard length	Standard length of predator
Predator total length	Total length of predator

Predator TL/FL/SL conversion reference	Reference for conversion between length dimensions
Standardized predator length	TL where available else SL or FL
Predator mass	Mass of predator
Predator mass unit	g
Predator mass check	Calculated mass of sphere with diameter = "Standardised predator length"
Predator mass check diff	Calculated mass of sphere with diameter = "Standardised predator length" – SI predator mass
Predator ratio mass/mass	Calculated mass of sphere with diameter = "Standardised predator length" / SI predator mass
SI predator mass	Predator mass (measured or calculated from length–mass regression)
Diet coverage	Types of prey items included in original work
Prey	Genus species name of prey
Prey common name	Common name of prey
Prey taxon	Ectotherm invertebrate, ectotherm vertebrate, egg, microzooplankton or mixed
Prey length	Prey length
Prey length unit	μm, mm, or cm
SI prey length	Prey length
Prey width	Prey width if measured
Prey width unit	μm or mm
Prey mass	Mass of prey
Prey mass unit	mg or g
Prey mass check	Calculated mass of sphere with diameter = "SI prey length"
Prey mass check diff	Calculated mass of sphere with diameter = "SI prey length" – SI prey mass (g)
Prey ratio mass/mass	Calculated mass of sphere with diameter = "SI prey length" / SI prey mass (g)

SI prey mass	Prey mass (measured or calculated from length–mass regression)
Geographic location	Location of study in original work
Latitude	Latitude of study
Longitude	Longitude of study
Depth	Estimated depth
Mean annual temp	Mean annual sea surface temperature
SD annual temp	Standard deviation annual sea surface temperature
Mean PP	Mean primary production
SD PP	Standard deviation primary production
Specific habitat	Description of location of study

III. Analysis of Data

III.1.1 Data Cleaning

In order to perform classification and regression on our dataset, we addressed data quality issues and made our dataset easier to work with.

III.1.1.1 Missing values

We first identified features with a significant amount of missing values by looking at the ratio of missing values to the total size of the dataset. After visualizing these ratios, we removed the sparse features in order to avoid inaccuracies or errors down the line. We ended up removing 'Individual ID', 'Predator TL/FL/SL conversion reference', 'Prey width', and 'Prey width unit' from our set of features.

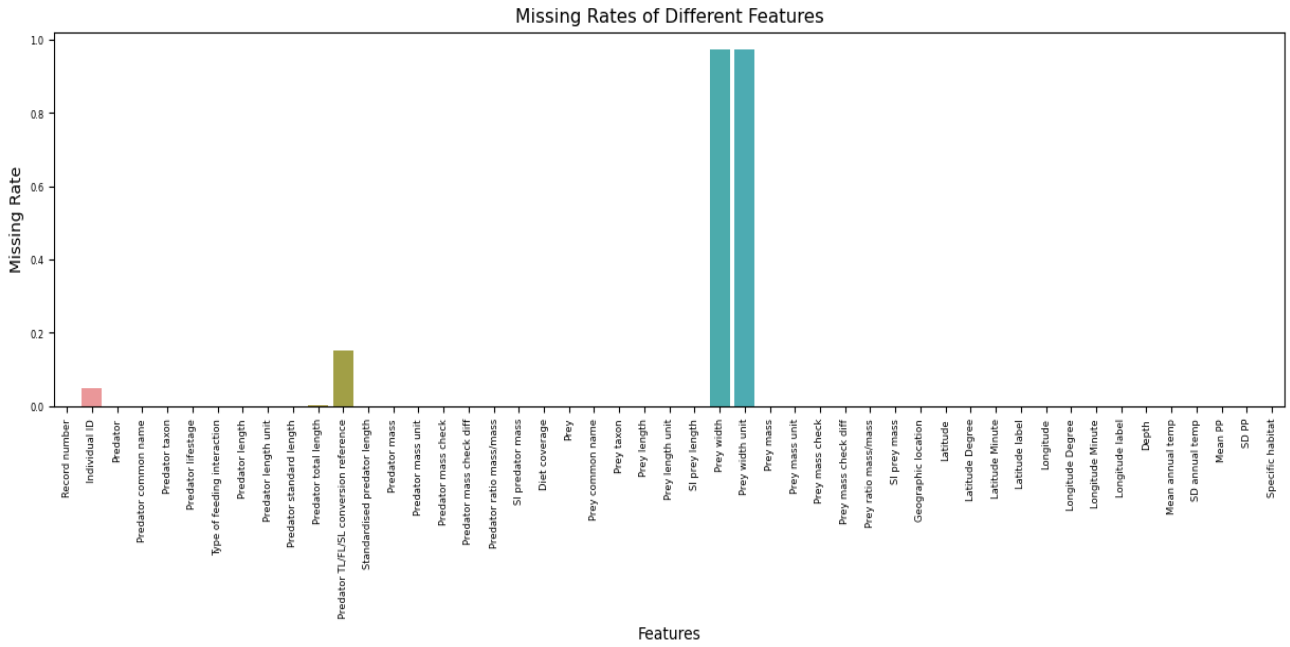


Figure 1: Bar plot of Missing values

III.1.1.2 Data Set Split

We then split the data into two - one for prey and another for predator - in order to perform more nuanced analyses on each. *'df_predator'* is the dataset aligned with the predators and *'df_prey'* is the dataset aligned with prey. They each contain a portion of unique data depending on whether the observation is a prey or predator, but because this was initially a single dataset, there are shared values. Each dataset contains features such as *'Geographic location'*, *'Specific habitat'* , and other unique spatial information about the given observation.

III.1.1.3 Unit Conversions

Finally, there was a lack of uniformity within the columns describing the length of an observation in both datasets due to a *'length'* column and a *'length unit'* column. For example, the *'length'* for two different observations could both say *'10'* but one could mean 10 mm and another be 10 centimeters - you'd have to consult the *'length unit'* column to find out. To avoid this extra step and allow for straightforward analysis, we normalized the predator and prey length columns according to their respective units column, using cm as a baseline unit. To do this, we simply multiplied each *'length'* column by their conversion rate according to the baseline unit. We encountered the same issue with mass as well (there's a *'mass'* column and a *'mass unit'* column), so we did the same procedure but instead using grams as the baseline unit. This then allowed us to do more accurate analysis, and also drop the *'length unit'* columns.

III.1.2 Exploratory Data Analysis

III.1.2.1 Distribution of dependent Variables

In order to get a better understanding of our dataset, we plot the hist plots of the dependent variables to see their distribution. The left graph shows the histogram of the predator common name. The right graph shows the histogram of prey taxon

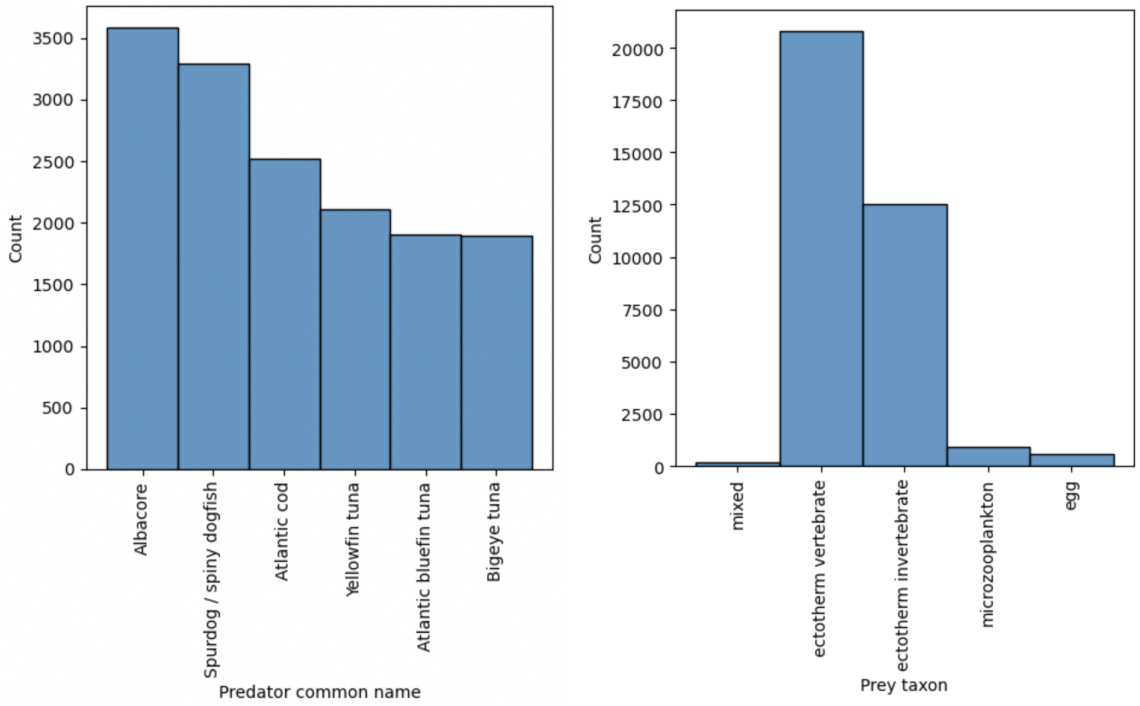


Figure 2: Hist plot of Dependent variables

III.1.2.2 Numerical Features

There are a lot of numerical features in our dataset, therefore, we want to visualize those features and see their distributions. There are two kinds of numerical features, the first kind is about the biological features of predators and prey, and the second kind is about the geographical features that are related to both predator and prey.

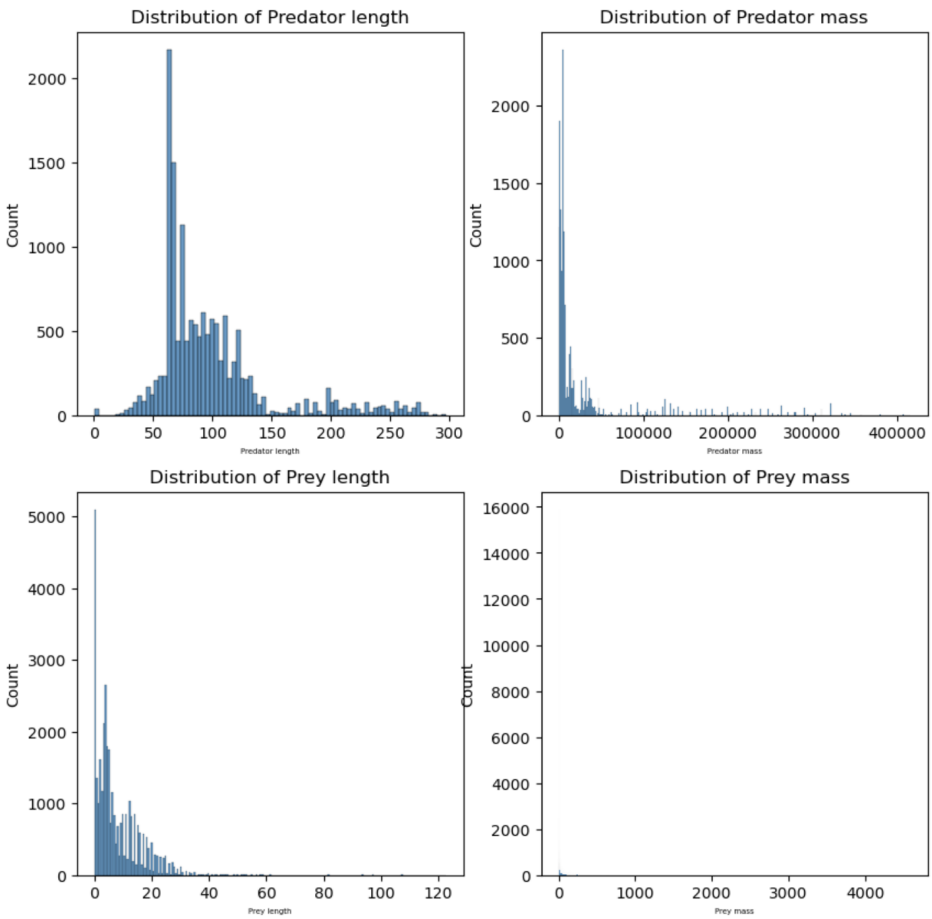


Figure 3: Hist plot of Biological features

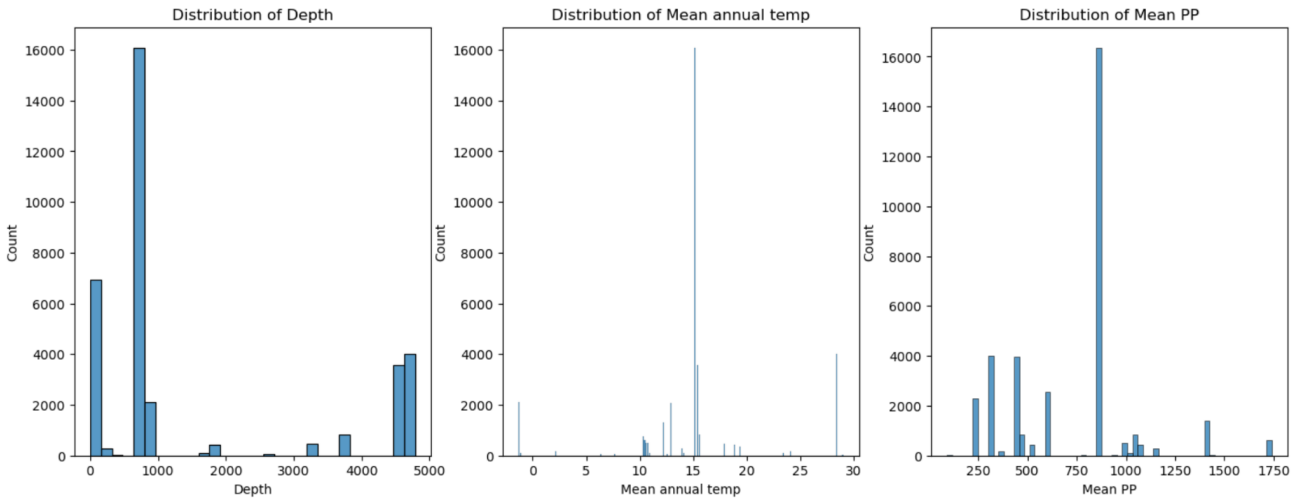


Figure 4: Hist plot of Geological features

III.1.2.3 Categorical Features

There are also a lot of categorical features in our dataset, and we want to visualize those features and see their distributions. We mainly focus on the distribution of the common names and taxon of predator and prey, since these features are closely related to the following machine learning analysis.

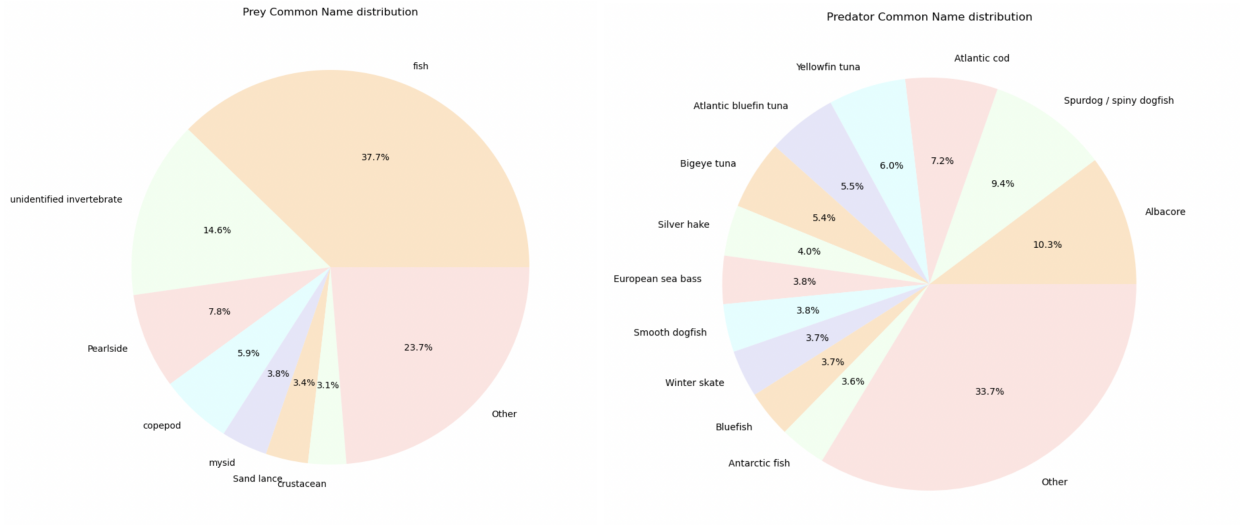


Figure 5: Pie Chart of Common names

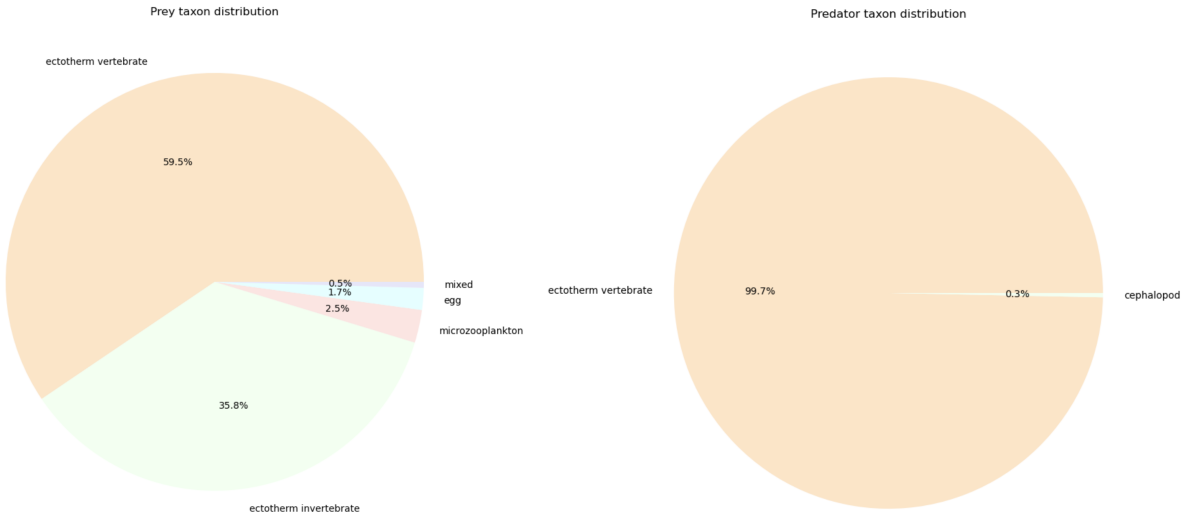


Figure 6: Pie Chart of Taxon

Two of the dependent variables are the common names of the predators and the taxon of prey. As you can see from the pie chart above, almost all of the prey is classified as one of the seven species. The rest don't have enough samples to be of use. This is also true for the predators where a lot of predators have very few samples. Therefore, we only use the majority groups for the further classification analysis.

IV. Model Building

IV.1 Classification Models

IV.1.1 Classify Predator Common Name

For the part of the dataset containing predators' biological features and geological features, we aim to figure out how well those features can help to classify the common names of predators. Our purpose is to build various classification models to predict the common name of a predator with its biological and geological characteristics, and compare the performances of all classification models we applied.

IV.1.1.1 Data preprocessing

Step one: Creating Y labels

There are a total of 87 common names in the dataset. By counting the values in each category, we find that most of the categories only have just one or two data points. In order to achieve a more accurate classification result, we calculate the ratio of the proportion of each category in the dataset, and only keep the records with a proportion higher than 0.05. After selecting the major groups of the marine species, we find that there are 6 common names in the new dataset. We use the common name of predators as our dependent variable and apply the characteristics of predators as independent variables to conduct further classification.

Step two: Categorical Variables Encoding

There are 6 categorical variables which are Predator taxon, Predator lifestage, Type of feeding interaction, Diet coverage, Geographic location, and Specific habitat. We use ordinal encoders to encode those categorical variables into numerical ones.

Step three: Train and Test Data Split

With the train set size : test set size equal to 0.2, we split our predator data into train dataset and test dataset for further classification.

Step four: Standard Scale

Standard Scaling which is normalization helps to increase the accuracy of classification models and can accelerate the training speed to get the optional solution. We fit and transform the train dataset and transform the test dataset.

IV.1.1.2 Methodology

We use SVC, Decision Tree, Random Forest, and Adaboost model to perform the classification. After training the model on the training date set, we calculate the train score and apply the model to the test set to calculate the test score which shows the performance of our model on a new dataset.

We first train the four classification models with the default hyperparameters. In order to get the best performance of our model, we use grid search with three fold cross validation to tune the hyperparameters of those four models. By printing out the best hyperparameters derived from the grid search, we train the best model of each method based on them.

In order to figure out what characteristics of predator influence the classification most, we pay attention to the feature importance of the four best models we trained and plot some bar charts to visualize the results. With the important features we acquire from our classification models, we can draw conclusions about the most important influential factors that influence the prediction of predators' common names.

IV.1.1.3 Model Performances

In our study we find that SVC has the best performance, followed by Decision Tree, Random Forest, and the least one is Adaboosting. The performance of models is shown in the table below.

Table 1: Predator Classification Model Performances

Models	**Default Model**		**Best Model**	
	Training Score	**Test Score**	**Best Hyperparameters**	**Best Mean Cross Validation Accuracy**
SVC	0.8931	0.8919	'C': 10, 'gamma': 1	0.98
Decision Tree	0.8951	0.8909	'criterion': 'entropy', 'max_depth': 6	0.95
Random Forest	0.9015	0.9026	'max_depth': 6, 'n_estimators': 10	0.92
Adaboost	0.4780	0.4817	'learning_rate': 0.5, 'n_estimators': 50	0.83

IV.1.1.4 Feature Importance

We get the paired feature importance of the four best classification models we trained and rank the features with the absolute value of their corresponding feature importance. We will show the top five most important features in the following content.

A. SVC:

('Predator lifestage', 0.2507300387024649),
('Mean PP', 0.2360341213344168),
('Depth', 0.23191475135497236),
('Predator length', 0.13364966017144195),
('Specific habitat', 0.11552890874742702)

B. Decision Tree:

('Mean annual temp', 0.4216168393869405),
('Mean PP', 0.38040805188666665),
('Predator mass', 0.10237907590256595),
('Predator length', 0.09559603282382692),
('Predator taxon', 0.0)

C. Random Forest:

('Specific habitat', 0.23146498866787268),
('Predator length', 0.1448286525969175),
('Mean PP', 0.14187334502847418),
('Geographic location', 0.12501059344637422),
('Predator mass', 0.10799359352813855)

D. Adaboost:

('Specific habitat', 0.26),
('Geographic location', 0.18),
('Mean PP', 0.18),
('Predator length', 0.16),
('Predator mass', 0.08)

We also plot the bar plots to visual the feature importance we get from our models.

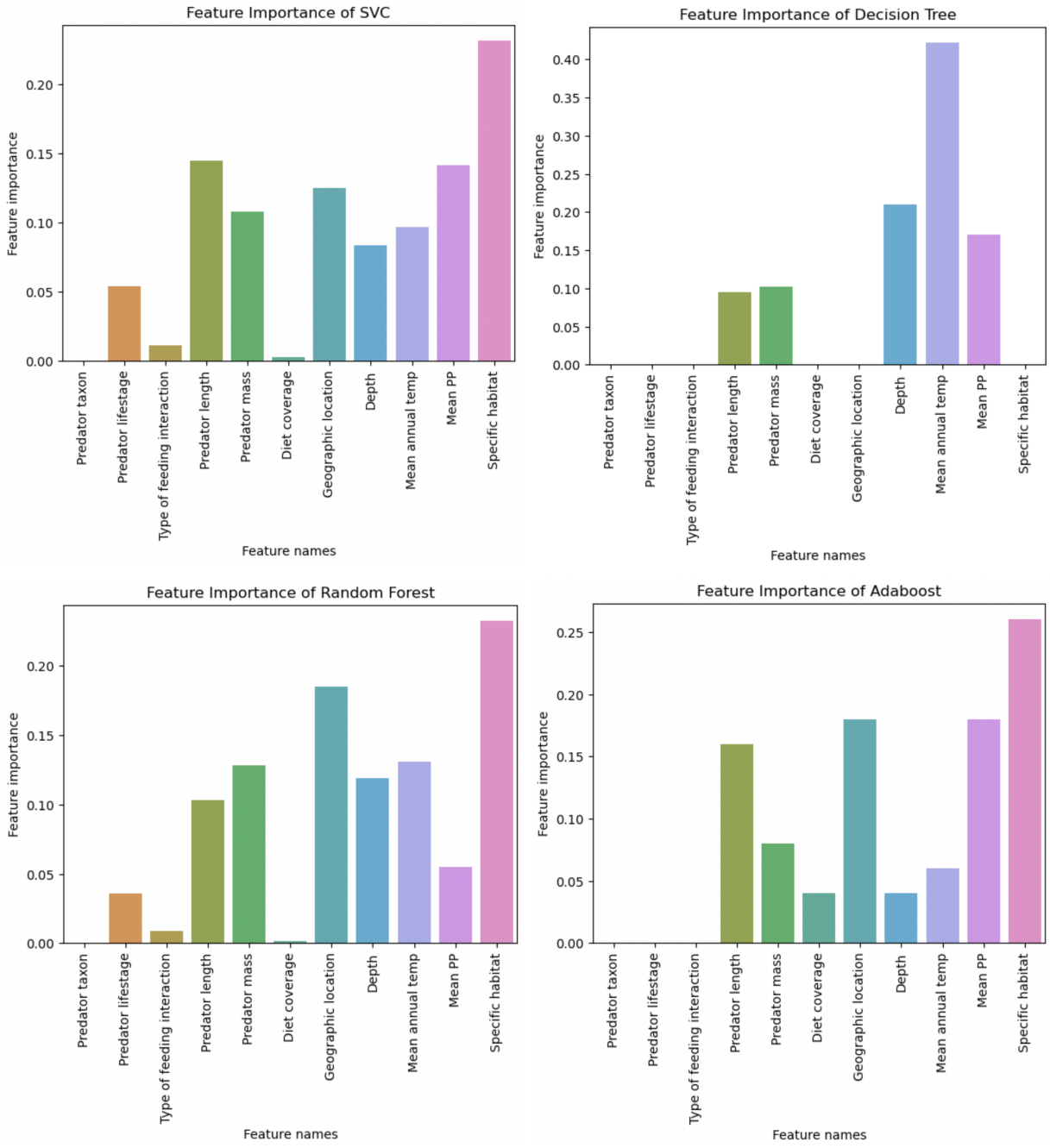


Figure 7: Feature Importance of Predator Classification

We can know from the graph that the outcome of Random Forest and Adaboost is kind of similar while the outcome of Decision is a little bit different compared to the other three models. Since Random Forest and Adaboost are all ensemble models, they are able to avoid overfitting and apply the majority votes as the final outcome of classification.

From the outcome we can conclude that “Specific habitat”, “Geographic location”, “Mean PP”, “Mean annual temp”, “Predator length”, and “Predator mass” are the significant characteristics that will be useful to help predict the common name of predator. So if there is a new marine animal without knowing the common name, we can use its length, mass and living places to predict its common name.

IV.1.2 Classify Prey Taxon

For the part of the dataset containing preys’ biological features and geological features, we aim to figure out how well those features can help to classify the taxon of preys. The purpose is the same as the predators' part: build various classification models to predict the taxon of a prey with its biological and geological characteristics, and compare the performances of all classification models we applied.

IV.1.2.1 Data preprocessing

Step one: Selecting Y labels

There are four types of taxon of preys in the data set. It seems like that it is a reasonable number that can be considered as a dependent variable. Meanwhile, it is important to note that predators can eat multiple types of prey, and these prey may have the same taxonomic classification. By using the living characteristics of prey as independent variables, we can still gain valuable insights into the potential factors that may influence the taxonomic classification of prey.

Step two: Categorical Variables Encoding

There are 2 categorical variables which are Geographic location, and Specific habitat. We use ordinal encoders to encode those categorical variables into numerical ones.

Step three: Train and Test Data Split

With the train set size : test set size equal to 0.2, we split our prey data into train dataset and test dataset for further classification.

Step four: Standard Scale

Standard Scaling which is normalization helps to increase the accuracy of classification models and can accelerate the training speed to get the optional solution. We fit and transform the train dataset and transform the test dataset.

IV.1.2.2 Methodology

We did the same thing that we did for predators. We trained four classification models, SVC, Decision Tree, Random Forest, and Adaboost, using prey taxon as the dependent variable and prey living characteristics as the independent variables. We calculated the train and test

scores to evaluate the performance of the models. Grid search with three-fold cross-validation was used to tune the hyperparameters of the models, and the best models were trained based on the best hyperparameters. The feature importance of the best models was analyzed, and bar charts were plotted to visualize the results. The important features were used to draw conclusions about the most influential factors that influence the prediction of prey taxon

IV.1.2.3 Model Performances

In our study we find that SVC has the best performance, followed by Decision Tree, Random Forest, and the least one is Adaboosting. The performance of models is shown in the table below.

Table 2 : Prey Classification Model Performances

Models	Default Model		Best Model	
	Training Score	Test Score	Best Hyperparameters	Best Mean Cross Validation Accuracy
SVC	0.8627	0.8606	'C': 10, 'gamma': 1	0.93
Decision Tree	0.8502	0.8513	'criterion': 'entropy', 'max_depth': 6	0.90
Random Forest	0.8487	0.8516	'max_depth': 6, 'n_estimators': 10	0.88
Adaboost	0.8022	0.8044	'learning_rate': 1, 'n_estimators': 10	0.80

IV.1.2.4 Feature Importance

We get the paired feature importance of the four best classification models we trained and rank the features with the absolute value of their corresponding feature importance. We will show the top three most important features in the following content.

A. **SVC:**
   ('Prey length', 2.374231651047012),
   ('Prey mass', 1.2307733769466722),
   ('Specific habitat', 0.37844675905171243)

B. **Decision Tree:**
   ('Prey length', 0.609117289870658),
   ('Specific habitat', 0.17913143158505582),
   ('Prey mass', 0.17496708325855898)

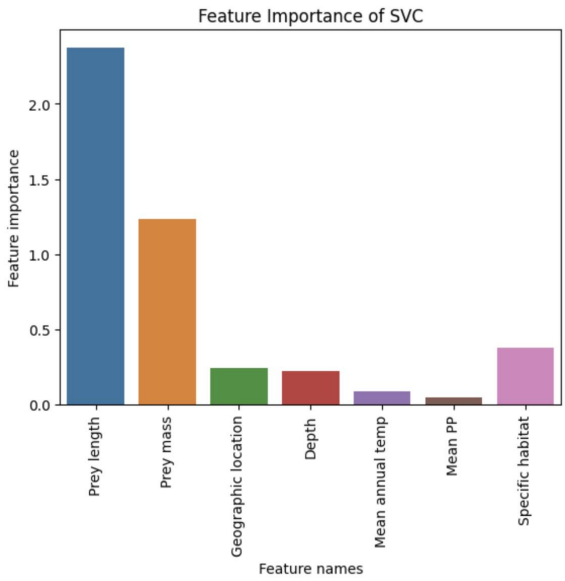
C. Random Forest:

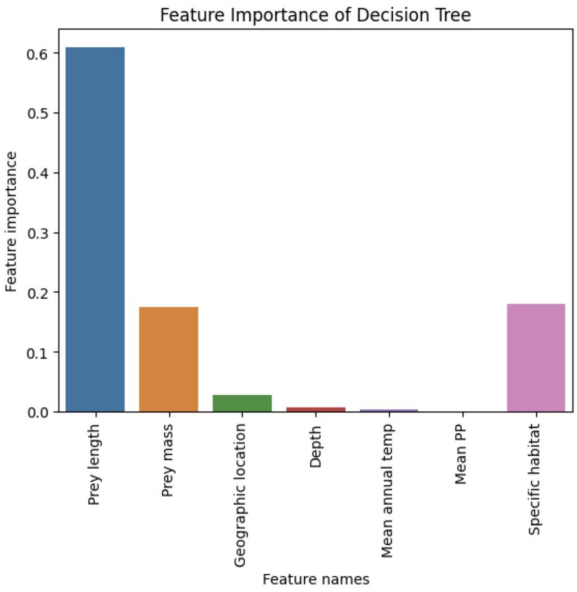
('Prey length', 0.3992603478326996),
('Prey mass', 0.18509756717898626),
('Mean annual temp', 0.12274348447250719)

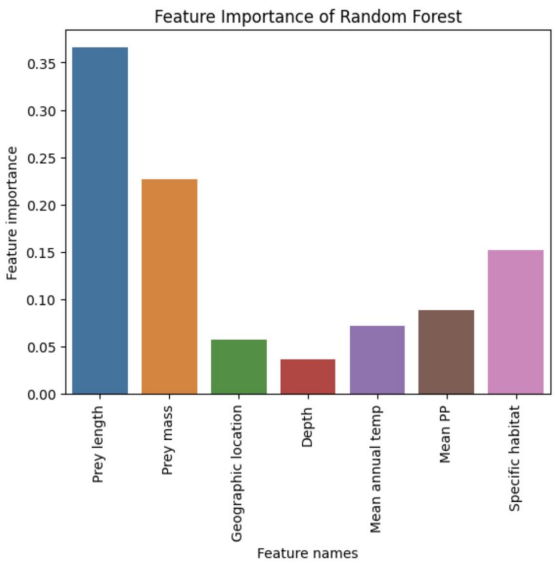
D. Adaboost:

('Specific habitat', 0.9),
('Prey length', 0.1),
('Prey mass', 0.0)

We plot some bar plots to visualize the feature importance that we get from our models.







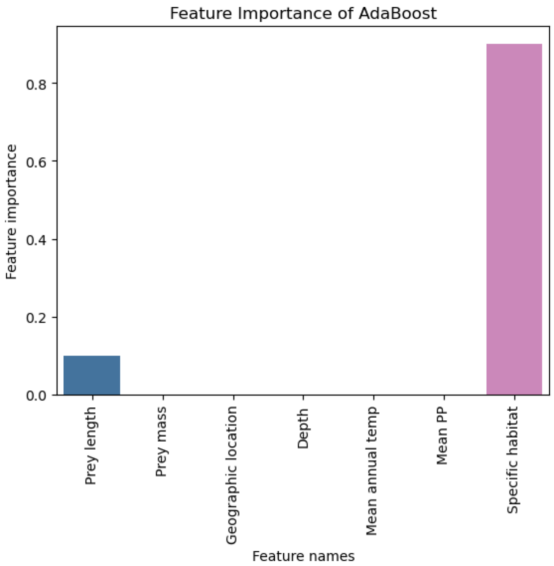


Figure 8: Feature Importance of Prey Classification

The bar plots of SVC and Random Forest show a similar pattern, suggesting that both algorithms have similar predictive power for the taxon of preys. On the other hand, the bar plot of AdaBoost shows some differences compared to the other two algorithms, indicating that it may perform differently in predicting the taxon of preys.

We can conclude that “Prey length”, “Prey mass”, and “Specific habitat” are the three most important factors that help us to predict the taxon of preys. So when we do not know the specific taxon of prey, we probably can utilize its length, mass, and living area to determine its taxonomic classification.

IV.2 PCA

In the classification part above, we find that the training time of models is quite long, especially in the grid search and cross validation process. The reason behind is that the number of features we used is large and there are a lot of columns to be considered. In order to address the problem of slow training speed and to pay more attention to the significant components that will mostly influence the machine learning model, we apply the PCA method.

We want to reduce the dataset to 3D using the PCA method. After conducting the PCA, we visualize the reduced dataset in a 3D scatter plot and also do the projection of those scatter points on the three dimensions. By using the labels of the dataset as a hue, we can clearly see from the graphs that PCA does help to reduce the data and in the meanwhile maintain the main information and attributes of the dataset.

The two figures below show the 3D reduced data of predator and prey (the left figure is the 3D visualization of predator and the right one is for prey):

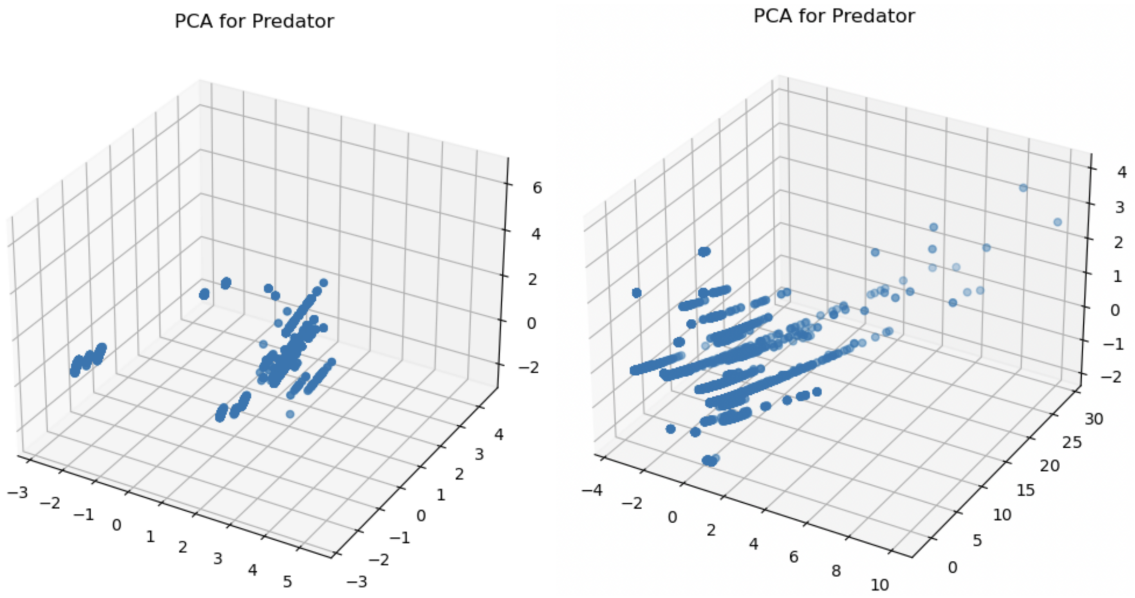


Figure 9: 3D scatter plot of the reduced data set

What's more, we plot the projection of the reduce data set on each dimension, which are shown in the figures below (the first one is for predator and the second one is for prey):

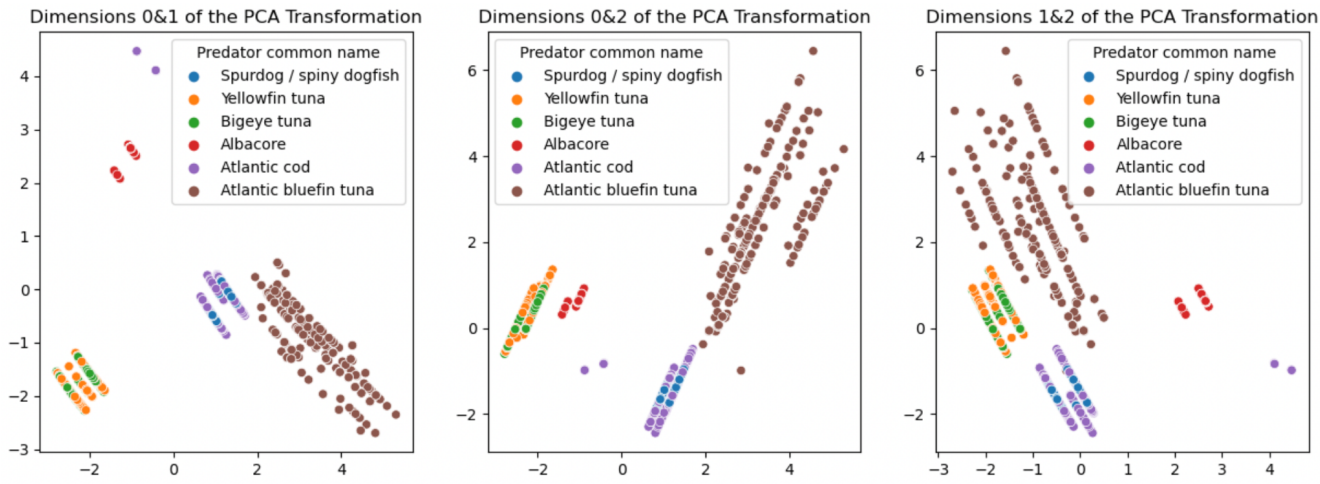


Figure 10: Scatter plot of PCA transformation on predator data

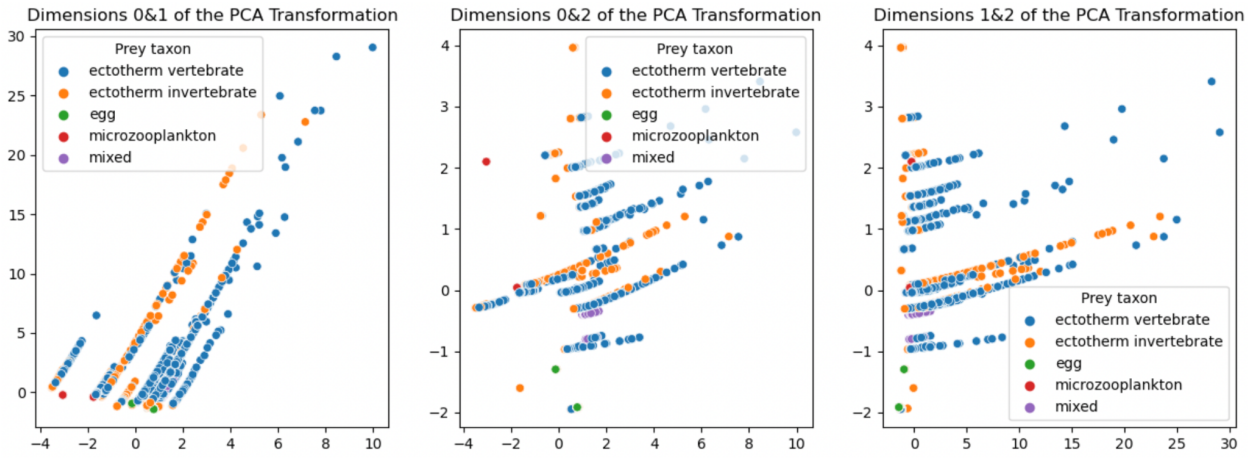


Figure 11: Scatter plot of PCA transformation on predator data

By applying PCA, we can reduce the data onto lower dimensions while still keeping the most important information they contain. PCA allows us to train our machine learning models at a faster speed. However, PCA transformation will lose part of the information of our original dataset. The missing information in the dimension reduction process will cause potential problems in the further machine learning model training process, such as increasing the errors. We applied the PCA outcome in the classification and regression but unfortunately we did not receive a better result. Therefore, we didn't use PCA transformed data in the following machine learning process.

IV.3 Kmeans

We learn from the classification above that predator dataset has 6 groups corresponding to their common names, and prey dataset has 5 groups corresponding to their taxon. Therefore, the two dataset we used have the clustering attributes. So we want to use the unsupervised clustering method we learned from lectures to evaluate the datasets.

Since K-means clustering is the unsupervised machine learning method, we only use features of predators and prey and ignore the category labels. In order to find the best dividition of the dataset, we take inertia into consideration.

As the number of groups increases the value of inertia will decrease as well, so the minimized inertia is not the target value we are looking for. As the number of groups becomes larger and larger, the decreasing speed of inertia varies. Therefore, we pay attention to the elbow and select the optimal number of groups according to it.

The two figures below show the trend of inertia of predator and prey (the left figure is the for predator and the right one is for prey):

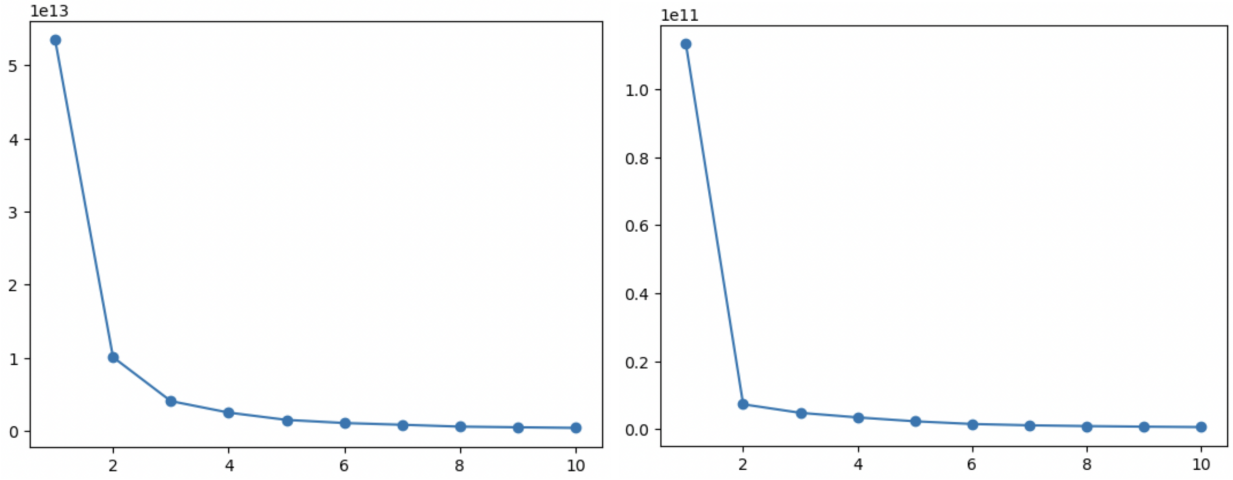


Figure 12: Line plot of Inertia trends

According to the figures, we set the number of groups equal to two for predators and the number of groups equal to two for prey as well. Then we use k-means to fit the dataset and predict the potential groups they belong to. By adding the original labels and the predicted labels onto the original dataset, we calculate the mean of predicted labels in each actual label group, aiming to evaluate model performance.

The two tables below in the screenshot show the outcome of k-means (the first one is for predator and the second one is for prey).

Table 3: Summary of K-means outcome for predator dataset

Predator common name	Predator taxon	Predator lifestage	Type of feeding interaction	Predator length	Predator mass	Diet coverage	Geographic location	Depth	Mean annual temp	Mean PP	Specific habitat	kmeans_label
Albacore	0.0	1.000000	0.336219	66.883552	6096.995309	0.000000	9.000000	4562.000000	15.400000	437.000000	1.000000	0.000000
Atlantic bluefin tuna	0.0	0.000000	0.123625	203.665584	156442.529544	0.000000	6.643793	56.707177	10.608853	1066.299633	3.000000	0.641173
Atlantic cod	0.0	0.040111	0.754170	66.486644	2691.241660	0.181493	0.209293	656.628674	14.879071	849.245036	2.114774	0.000000
Bigeye tuna	0.0	0.000000	0.804004	107.865121	14839.720179	0.000000	3.000000	4785.000000	28.400000	316.000000	6.000000	0.000000
Spurdog / spiny dogfish	0.0	0.000000	0.421053	82.239428	2547.490880	0.031031	0.020688	671.052327	15.077244	864.310618	2.020688	0.000000
Yellowfin tuna	0.0	0.000000	1.228585	108.681022	26764.666540	0.000000	3.000000	4785.000000	28.400000	316.000000	6.000000	0.000000

Table 4: Summary of K-means outcome for prey dataset

Prey taxon	Prey length	Prey mass	Geographic location	Depth	Mean annual temp	Mean PP	Specific habitat	kmeans_label
ectotherm invertebrate	3.560657	13.905011	8.013103	1599.555928	14.518936	726.588607	8.148929	0.740812
ectotherm vertebrate	11.430129	39.217246	7.630517	1531.068544	15.756414	739.622286	6.959471	0.748833
egg	0.030330	0.000002	5.080944	771.573356	1.017032	509.966273	7.843170	0.983137
microzooplankton	0.060302	0.000008	14.585779	2559.448081	18.395485	562.847630	16.504515	0.495485
mixed	11.351194	21.888489	4.559006	14.006211	23.652174	866.000000	0.000000	1.000000

We find that K-means perform better on the predator dataset than prey dataset, probably because predator dataset contains more features than prey dataset, so that K-means can take more attributes into consideration and generate a better result.

There are some insights we can get from the dataset:

1. When clustering into two groups, the predator dataset is separated into bluefin tuna and others. By looking for some background information we learn that the weight of bluefin tuna is much larger than the other, and also the living depth of bluefin tuna is significantly different from others. That is why K-means clustering separate bluefin tuna as a sub-group and group others in a group.
2. Prey with a mixed taxon has the living depth significantly different from others, so it can be easily separated. Due to the similarity of attributes of prey in different categories, K-means did not show a great performance in clustering prey.

IV.4 Regression

For the regression task, we were interested in being able to predict some quantitative difference between the predator-prey relationship pair. Before with classifications, we broke the relationship in order to analyze the predators and the prey separately, but by recombining the two, we can make predictions for the relationship between the two. Our overarching goal for the regression problem was to understand how the various geographic features (defined as any feature not explicitly related to the predator or the prey) can perform when tasked with predicting some predator-prey relationship quantity.

IV.4.1 Data Preparation

Due to the nature of our goal, we had to first recombine the predator and prey data. Since we did not drop any rows when we cleaned our data for the classification, we can simply take the same dataframes and join on their common index. Next, we performed categorical feature encoding. We took all of the columns that were not integers or floats, and used an ordinal encoder in order to make the data numerical and thus passable to our models. Lastly, we split our dataset into testing and training data, and scaled the data using a standard scalar.

IV.4.2 Predicting difference in Predator-Prey mass using Geographic Features

Our first goal was to predict the difference between the predator and prey masses using the geographical features such as mean water temperature, latitude and longitude, depth, and so forth. To evaluate whether these geographical features could be a predictor for the difference between predator and prey masses, we performed six different regression models (Linear Regression, Ridge Regression, Lasso Regression, Decision Tree Regression, AdaBoost Regression, Random Forest Regression).

IV4.2.1 Model Performance

Using the same test train methodology as the classifier, and performing the same hyperparameter tuning, we get the following results.

Table 5 : Mass Regression Model Performances

Models	Default Model		Best Model	
	Training Score	Test Score	Best Hyperparameters	Best Mean Cross Validation Accuracy
Linear	0.5809	0.5563	—	—
Ridge	0.5808	0.5562	‘alpha’: '0.1’	0.58
Lasso	0.5787	0.5532	‘alpha’: '0.01’	0.58
Decision Tree	0.7479	0.7490	‘criterion’: ‘mse’, ‘max_depth’: 6	0.74
Adaboost	0.5800	0.5931	'learning_rate': 0.01, 'n_estimators': 150	0.71
Random Forest	0.7479	0.7490	‘max_depth’: 6, ‘n_estimators’: 10	0.74

IV4.2.2 Feature Importance

We get the paired feature importance of the three best classification models we trained and rank the features with the absolute value of their corresponding feature importance. We will show the top three most important features in the following content.

A. Random Forest
(Latitude Minute, 0.5857)
(Geographic Location, 0.1000)
(Mean Annual Temp, 0.0914)

B. Decision Tree
(Latitude Minute, 0.6729)
(Mean Annual Temp, 0.1825)
(Geographic Location, 0.0794)

C. Adaboost
(Geographic Location, 0.3181)

(Latitude Minute, 0.2771)
(Latitude, 0.2021)

Plotting the six most important features for each of our three best performing models, we see the following graphs.

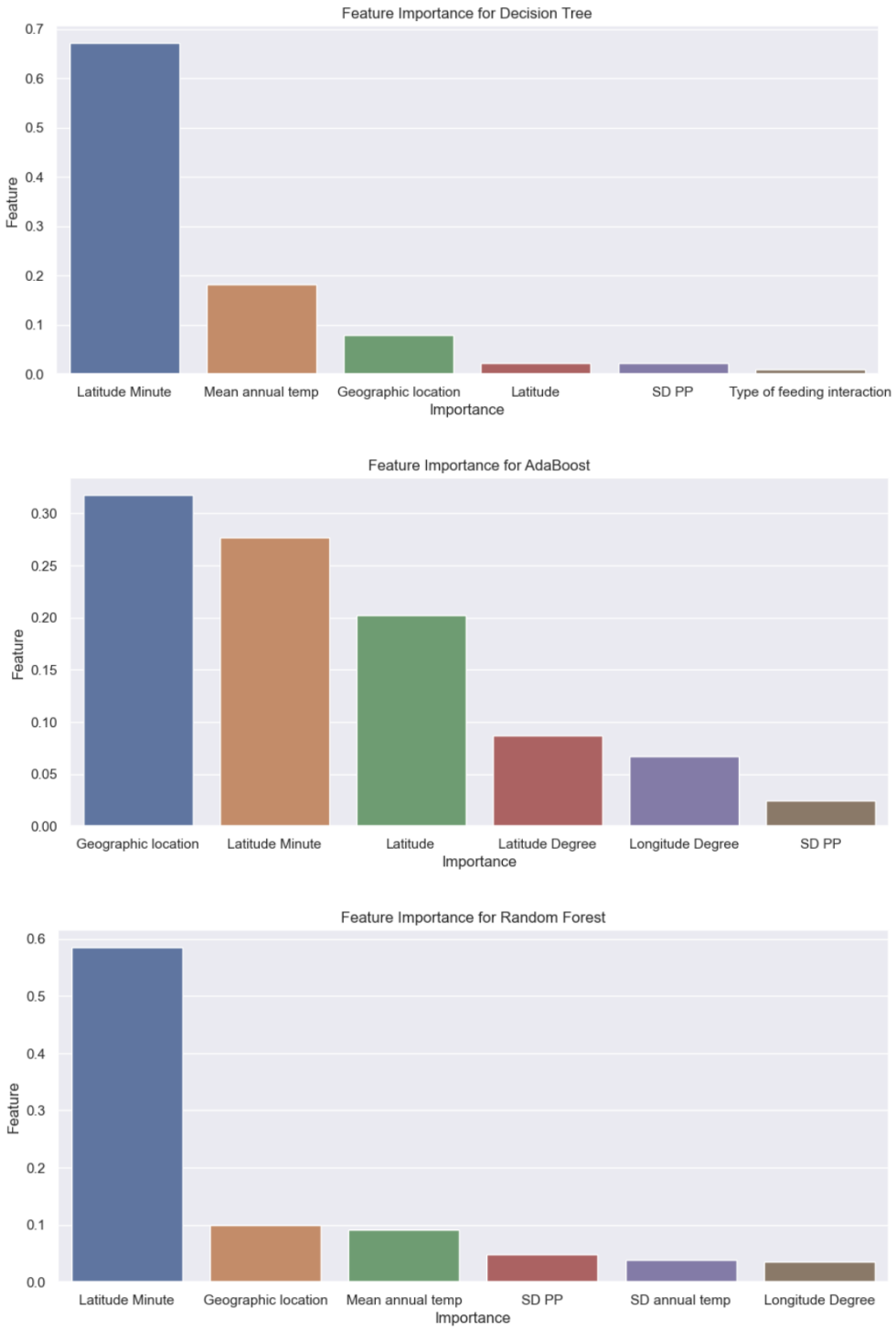


Figure 13: Feature Importance of Mass Regression

Since Latitude Minute and Geographic Location were features that were in the top three feature importance considerations for all three of our best models, we can deduce that those two features are instrumental in being able to predict the difference in Predator-Prey mass differences. What is especially interesting is that in our best model, Random Forest, we see that Latitude Minute is a clear dominant winner in terms of feature importance.

IV.4.3 Predicting difference in Predator-Prey length using Geographic Features

Our second goal was to predict the difference between the predator and prey lengths using the geographical features such as mean water temperature, latitude and longitude, depth, and so forth. To evaluate whether these geographical features could be a predictor for the difference between predator and prey lengths, we performed six different regression models (Linear Regression, Ridge Regression, Lasso Regression, Decision Tree Regression, AdaBoost Regression, Random Forest Regression).

IV.4.3.1 Model Performance

Using the same test train methodology as the classifier, and performing the same hyperparameter tuning, we get the following results.

Table 6 : Length Regression Model Performances

Models	**Default Model**		**Best Model**	
	Training Score	**Test Score**	**Best Hyperparameters**	**Best Mean Cross Validation Accuracy**
Linear	0.7123	0.6992	—	—
Ridge	0.7122	0.6990	‘alpha’: '0.1’	0.71
Lasso	0.6267	0.6125	‘alpha’: '0.01’	0.71
Decision Tree	0.7903	0.7860	‘criterion’: ‘mse’, 'max_depth': 6	0.79
Adaboost	0.7678	0.7615	'learning_rate': 0.5, 'n_estimators': 50	0.77
Random Forest	0.7903	0.7860	‘n_estimators’: 100, 'max_depth': 6	0.79

IV.4.3.2 Feature Importance

We get the paired feature importance of the three best classification models we trained and rank the features with the absolute value of their corresponding feature importance. We will show the top three most important features in the following content.

A. Random Forest

(Latitude Minute, 0.3287)
(Depth, 0.2152)
(Longitude Degree, 0.1104)

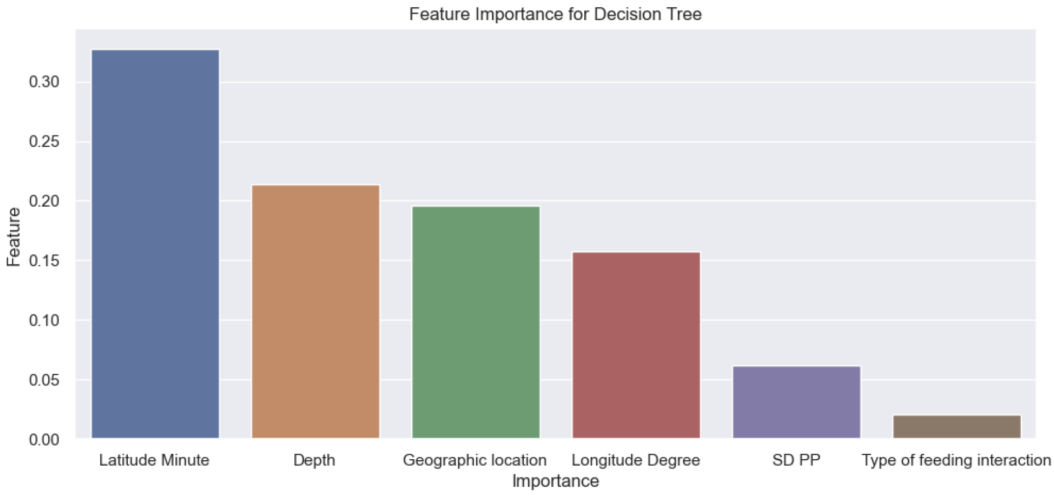
B. Decision Tree

(Latitude Minute, 0.3278)
(Depth, 0.2141)
(Geographic Location, 0.1956)

C. Adaboost

(Mean Annual Temp, 0.2065)
(Longitude Minute, 0.1615)
(SD Annual Temp, 0.1389)

Plotting the six most important features for each of our three best performing models, we see the following graphs.



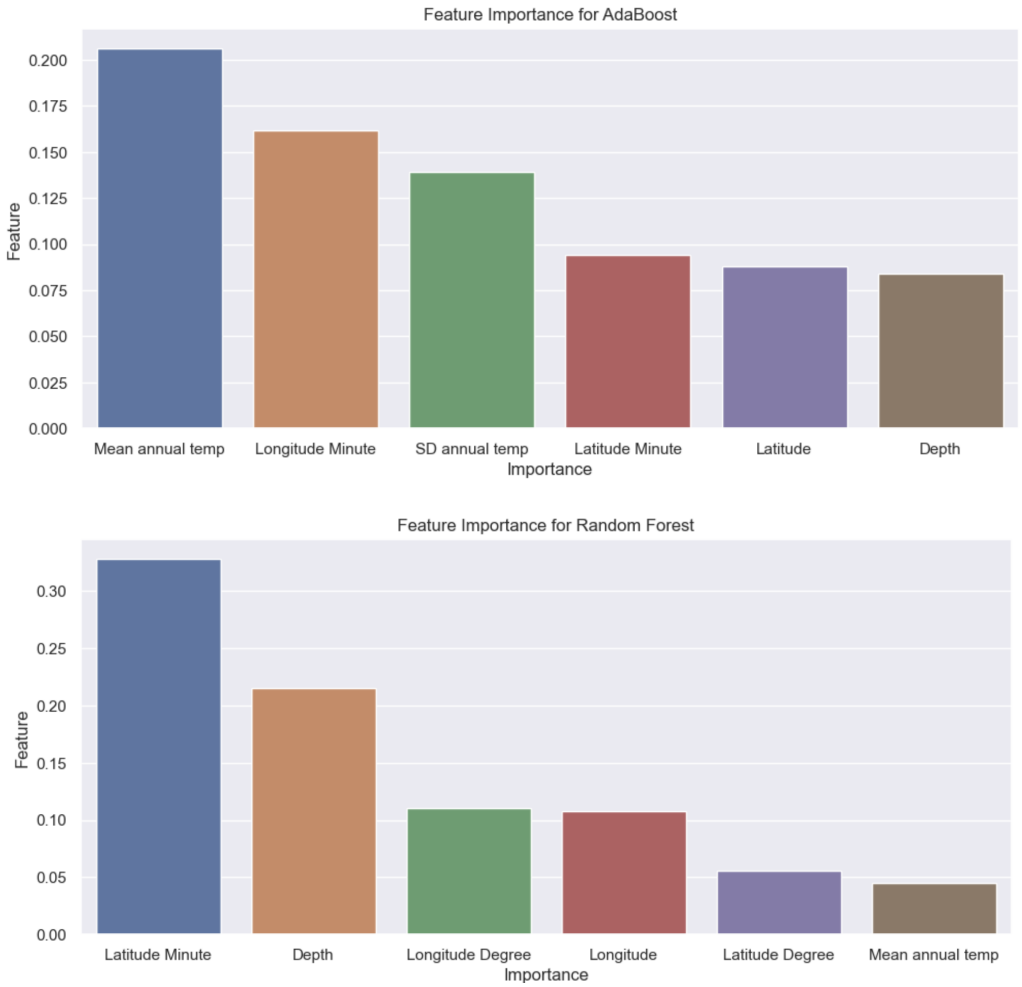


Figure 14: Feature Importance of Length Regression

What is interesting about the feature importance extraction task for predicting Predator-Prey length differences is that we do not see a certain subset of features dominate across all models. Instead, we find that each model uses different features in order to come to the same conclusion in terms of training and testing score. Latitude minute is the best feature that we see in two of our models, yet it is absent from the top 3 features of our third best performing model.

V. Discussion

From the classification, the models are able to predict the predator's name with very high accuracy. The main concern when getting a very high accuracy is overfitting. There are a few reasons overfitting is not a concern in this case. First of all, the models are trained on six independent variables. Second, the model is only trying to classify six different species of predators. The features used in these models including the length, mass and habitat of the animal are some ways that wildlife biologists are able to classify sea life in the real world.

From the clustering, the model groups the data into a few larger groups and does not perform so well on clustering the data into the numbers of their original groups. The reason behind is that we do not have a sufficient number of features of the species we analyzed, with the current attributes, it's hard to correctly group the species. Although the labels of predators are different, they are at the same larger level in biology so the difference between them is slight. In order to cluster them better, we would like to have more biological information about the marine species and build a deeper understanding about them.

From the regression analysis, we find a new way to predict the biological features of marine species by using the information about their living environment. It's a great way to understand marine species without knowing the detailed information about the species itself. However, we still need to improve the performance of our regression models by including more features that we can get in the future.

The limitations of our analysis is that although there are several columns of our dataset, a lot of columns are similar to each other without providing much additional information. Therefore, the information we can use for machine learning analysis is limited. Also, there is not so much relational information about predators and prey so it's hard for us to analyze the biological chain. In order to better understand the predator and prey in marine environments, we need to look for more correlated data and then carry out a more comprehensive analysis.

Understanding the marine species has significant meaning since it can help to know which kind of marine environment is suitable for which kind of species to live, and we can use that to better protect the life of marine species. Also, by knowing the biological chain of marine environments, we can better understand the biological mechanism and maintain a healthier state of marine ecosystem.

VI. Conclusion

This project involves using classification models, such as SVC, Decision Tree, Random Forest, and Adaboost, to predict the common name of predators and the taxon of prey. Based on the analysis, it has been determined that specific habitat, geographic location, mean primary productivity, mean annual temperature, predator length, and predator mass are the most important features in predicting the common name of predators. On the other hand, prey length, prey mass, and specific habitat are identified as the three most significant factors in predicting the taxon of prey.

PCA was utilized to reduce the dataset to 3D and prioritize the most significant features. The reduced dataset was then depicted in a 3D scatter plot with the data points projected on each of the three dimensions. However, since PCA transformed data did not yield a superior result and was therefore not used.

K-means clustering method was used to evaluate the predator and prey datasets in the clustering section. The number of groups was determined based on the elbow method of the inertia graph, which was 2. The predator dataset yielded better results than the prey dataset, potentially because it had more features. In the predator dataset, bluefin tuna was separated as a sub-group, while prey with a mixed taxon was easily separated due to differences in living depth.

We apply the linear regression model to analyze whether there is a relationship between predator characteristics and prey length and mass. The datasets were combined and encoded, split into training and testing sets, and scaled before applying the regression models. The key features for predicting prey mass were found to be Latitude Minute and Geographic Location, while Latitude minute was found to be important for predicting prey length.

In conclusion, analyzing the relationship between predator and prey size is crucial for understanding the dynamics of ecosystems and food webs. By examining predator and prey size, we can better understand the structure and functioning of food webs, as well as the ecological impacts of changes in predator and prey populations. This information can also inform management and conservation efforts to maintain healthy and balanced ecosystems.

VII. Appendix

VII.1 Group Contributions

We work together toward the entire project, and everyone is working hard on analyzing the problem and getting some insights. We held several meetings to discuss our goal to achieve and what kinds of machine learning methods we should apply based on our dataset. Also we make the decision together that we should focus on all the classification, clustering, and regression tasks since they are all suitable for our dataset.

For detailed contributions:

- Jacob and Nixon are responsible for the dataset choosing, data analysis, data cleaning and preprocessing, EDA parts. They clean the dataset, handle the missing values, unify the format of data, and make plots etc.
- Jingyi and Xinyue work together on the preprocessing of machine learning, including encoding, feature selection, wrangling the data frame etc.
- Jingyi, Xinyue and Yajie are working on the machine learning parts. Jingyi conducts the classification of predators and Xinyue conducts the classification of prey, Yajie is responsible for the regression analysis. Also, Jingyi analyzes the PCA and Clustering
- For report writing, everyone is responsible for writing the part they analyze. Xinyue organizes the report and completes the abstract, introduction, and conclusion part.

VII.2 Copy of Dataset

The link of dataset is: https://github.com/JingyiZhang0129/5241_project

The cleaned dataset we use is the the dataset with the name “5241dataset”, the original dataset we downloaded from the website is the dataset with the name “Predator_and_prey”

VII.3 Copy of Code

Please see these in the following pages.

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.model_selection import validation_curve
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso

from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC

from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

In [ ]:
```
# Load Data
df = pd.read_csv("5241dataset.csv", encoding = 'unicode_escape')
```

1. Dataset Description

In [ ]:
```
df.info()
```

In [ ]:
```
df.describe().T
```

2. Data Cleaning

2.1 Missing Values

In [ ]:
```
col_na = df.columns[df.isna().any()]
col_na_ratio = df.isna().sum()/df.shape[0]

print("The columns contain missing values are:\n", col_na)

fig,ax = plt.subplots(figsize=(15,4))
sns.barplot(x = df.columns, y = col_na_ratio, ax = ax)

ax.tick_params(axis = 'x', rotation = 90)
ax.set_xlabel('Features')
ax.set_ylabel('Missing Rate')
ax.set_title('Missing Rates of Different Features')

plt.tick_params(labelsize=6)
plt.show()
```

In [ ]:
```
df.drop(['Individual ID','Predator TL/FL/SL conversion reference','Pre
```

In [ ]:
```
df.columns
```

2.2 Dataset Split

In [ ]:
```
df_predator = df[['Record number', 'Predator', 'Predator common name',
                  'Predator lifestage', 'Type of feeding interaction', '
                  'Predator length unit', 'Predator standard length','Pr
                  'Standardised predator length','Predator mass', 'Preda
                  'Predator mass check diff', 'Predator ratio mass/mass'
                  'Geographic location', 'Latitude', 'Latitude Degree',
                  'Longitude', 'Longitude Degree', 'Longitude Minute','L
                  'Mean annual temp', 'SD annual temp','Mean PP', 'SD PP
df_prey = df[['Prey', 'Prey common name', 'Prey taxon', 'Prey length',
                  'Prey mass', 'Prey mass unit', 'Prey mass check', 'Pre
                  'SI prey mass', 'Geographic location', 'Latitude', 'La
                  'Latitude label', 'Longitude', 'Longitude Degree', 'Lo
                  'Depth', 'Mean annual temp', 'SD annual temp','Mean PP
```

2.2.1 Predator Dataframe

In [ ]:
```
df_predator['Predator length unit'].value_counts()
```

In [ ]:
```
for i in range(df_predator.shape[0]):
    if df_predator['Predator length unit'][i] == 'mm':
        df_predator['Predator length'][i] = df_predator['Predator leng
    if df_predator['Predator length unit'][i] == 'µm':
        df_predator['Predator length'][i] = df_predator['Predator leng
    else:
        continue
```

In [ ]:
```
df_predator['Predator mass unit'].value_counts()
```

In [ ]:
```
df_predator.drop(['Predator length unit'], axis = 1, inplace = True)
```

In [ ]:
```
df_predator['Predator common name'].value_counts()
```

In [ ]:
```
major_predator = df_predator['Predator common name'].value_counts()/df
major_predator_names = major_predator[major_predator == True].index
major_predator_names
```

In [ ]:
```
df_names = []
for name in major_predator_names:
    df_names.append(df_predator[df_predator['Predator common name'] ==
df_predator_new = pd.concat(df_names, ignore_index= True)
```

In [ ]:
```
predator_y = df_predator_new['Predator common name']
predator_x = df_predator_new[['Predator taxon','Predator lifestage', '
                              'Diet coverage','Geographic location','Dep
```

2.2.2 Prey Dataframe

In [ ]:
```
df_prey['Prey length unit'].value_counts()
```

In [ ]:
```
for i in range(df_prey.shape[0]):
    if df_prey['Prey length unit'][i] == 'mm':
        df_prey['Prey length'][i] = df_prey['Prey length'][i]*0.1
    if df_prey['Prey length unit'][i] == 'µm':
        df_prey['Prey length'][i] = df_prey['Prey length'][i]*0.0001
    else:
        continue
```

In [ ]:
```
df_prey['Prey mass unit'].value_counts()
```

In [ ]:
```
for i in range(df_prey.shape[0]):
    if df_prey['Prey mass unit'][i] == 'mg':
        df_prey['Prey mass'][i] = df_prey['Prey mass'][i]*0.001
    else:
        continue
```

In [ ]:
```
df_prey.drop(['Prey length unit','Prey mass unit'], axis = 1, inplace
```

In [ ]:
```
prey_y = df_prey['Prey taxon']
prey_x = df_prey[['Prey length','Prey mass', 'Geographic location','De
```

3. Data Visualization

3.1 Correlations of Independent Variables

In [ ]:
```
# Predators
corr_matrix = predator_x.corr()
fig = plt.figure(figsize = (4,4))
plt.title('Correlations of Predators')
sns.heatmap(corr_matrix, annot=True, annot_kws={"size":6}, center=0, c
plt.show()
```

In [ ]:
```
# Preys
corr_matrix = prey_x.corr()
fig = plt.figure(figsize = (4,4))
plt.title('Correlations of Preys')
sns.heatmap(corr_matrix, annot=True, annot_kws={"size":6}, center=0, c
plt.show()
```

In [ ]:
```
df['Prey taxon'].unique()
```

3.2 Distribution of Dependent Variable

In [ ]:
```
fig, ax=plt.subplots(figsize=(5,5))
sns.histplot(predator_y, ax=ax)
plt.xticks(rotation = 90)
plt.show()
```

In [ ]:
```
fig, ax=plt.subplots(figsize=(5,5))
sns.histplot(prey_y, ax=ax)
plt.xticks(rotation = 90)
plt.show()
```

3.3 Numerial Features

In [ ]:
```
#predator and prey
df_v = pd.concat([predator_x,prey_x],axis=1)

numeric_cols = ['Predator length', 'Predator mass','Prey length', 'Pre
print(len(numeric_cols))

fig, ax=plt.subplots(nrows=2, ncols=2, figsize=(10,10))

for var, subplot in zip(numeric_cols, ax.flatten()):
    b=sns.histplot(x=var, data=df_v, ax=subplot)
    b.set_xlabel(str(var), fontsize = 5)
    b.set_title(f'Distribution of {var}')

plt.show()
```

In [ ]:
```
# geographical
df_v = pd.concat([predator_x,prey_x],axis=1)

numeric_cols = ['Depth', 'Mean annual temp','Mean PP']
print(len(numeric_cols))

fig, ax=plt.subplots(nrows=1, ncols=3, figsize=(17,6))

for var, subplot in zip(numeric_cols, ax.flatten()):
    b=sns.histplot(x=var, data=df, ax=subplot)
    b.set_xlabel(str(var), fontsize = 10)
    b.set_title(f'Distribution of {var}')

plt.show()
```

3.4 Categorical Features

In [ ]:
```
# Prey Common Names
def myformat(value):
    return f'{value:.1f}%'

threshold = 1000
y = df['Prey common name'].value_counts()
mylabels =y.index
print(mylabels[y >= threshold])
small_values = y[y < threshold]
other_value = sum(small_values)
y = y[y >= threshold]
mylabels = y.index
y = np.append(y, other_value)
mylabels = np.append(mylabels, "Other")

fig1, ax1 = plt.subplots(figsize = (5,5))
ax1.pie(y, labels=mylabels, textprops={'rotation': 0}, autopct=myforma
ax1.set_title("Prey Common Name distribution")
plt.show()
```

In [ ]:
```
# Prey taxon
def myformat(value):
    return f'{value:.1f}%'

threshold = 1000
y = df['Prey taxon'].value_counts()
mylabels =y.index

fig1, ax1 = plt.subplots(figsize = (5,5))
ax1.pie(y, labels=mylabels, textprops={'rotation': 0}, autopct=myforma
ax1.set_title("Prey taxon distribution")
plt.show()
```

In [ ]:
```
# predator common names
def myformat(value):
    return f'{value:.1f}%'

threshold = 1000
y = df['Predator common name'].value_counts()
mylabels =y.index
print(mylabels[y >= threshold])
small_values = y[y < threshold]
other_value = sum(small_values)
y = y[y >= threshold]
mylabels = y.index
y = np.append(y, other_value)
mylabels = np.append(mylabels, "Other")

fig1, ax1 = plt.subplots(figsize = (5,5))
ax1.pie(y, labels=mylabels, textprops={'rotation': 0}, autopct=myforma
ax1.set_title("Predator Common Name distribution")
plt.show()
```

In [ ]:
```
# Predator taxon
def myformat(value):
    return f'{value:.1f}%'

threshold = 1000
y = df['Predator taxon'].value_counts()
mylabels =y.index

fig1, ax1 = plt.subplots(figsize = (5,5))
ax1.pie(y, labels=mylabels, textprops={'rotation': 0}, autopct=myforma
ax1.set_title("Predator taxon distribution")
plt.show()
```

4. Classification Model

4.1 Classify Predator common names

In [ ]:
```
# Categorical Feature Encoding
ordinalencoder = OrdinalEncoder()
predator_x['Predator taxon'] = ordinalencoder.fit_transform(predator_x
predator_x['Predator lifestage'] = ordinalencoder.fit_transform(predat
predator_x['Type of feeding interaction'] = ordinalencoder.fit_transfo
predator_x['Diet coverage'] = ordinalencoder.fit_transform(predator_x[
predator_x['Geographic location'] = ordinalencoder.fit_transform(preda
predator_x['Specific habitat'] = ordinalencoder.fit_transform(predator
```

In [ ]:
```
# Train Test Split
X_train_predator, X_test_predator, y_train_predator, y_test_predator =




# Scaling
scaler = StandardScaler()
X_train_predator = scaler.fit_transform(X_train_predator)
X_test_predator = scaler.transform(X_test_predator)
```

4.1.1 SVC

In [ ]:
```
svc = SVC(C=1,gamma='scale')
svc.fit(X_train_predator,y_train_predator)
print(f"The train score is:",svc.score(X_train_predator,y_train_predat
print(f"The test score is:",svc.score(X_test_predator,y_test_predator)
```

In [ ]:
```
# Grid Search
params = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01]}
svc_gscv = GridSearchCV(estimator = SVC(random_state=123), param_grid=
svc_gscv.fit(X_train_predator, y_train_predator)
print(f'svc best hyperparams      : {svc_gscv.best_params_}')
print(f'svc best mean cv accuracy : {svc_gscv.best_score_:.2f}')
```

In [ ]:
```
svc_best = SVC(C=10,gamma=1,kernel='linear')
svc_best.fit(X_train_predator,y_train_predator)
```

In [ ]:
```
importances = svc_best.coef_[0]
features_dict = dict(zip(predator_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=predator_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of SVC")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.1.2 Decision Tree

In [ ]:
```
dt = DecisionTreeClassifier(criterion='gini', max_depth = 5)
dt.fit(X_train_predator,y_train_predator)
print(f"The train score is:",dt.score(X_train_predator,y_train_predato
print(f"The test score is:",dt.score(X_test_predator,y_test_predator))
```

In [ ]:
```
# Grid Search
params = {'criterion': ['gini', 'entropy'], 'max_depth' : [2,3,4,5,6]}
dt_gscv = GridSearchCV(estimator = DecisionTreeClassifier(random_state
dt_gscv.fit(X_train_predator, y_train_predator)
print(f'dt best hyperparams      : {dt_gscv.best_params_}')
print(f'dt best mean cv accuracy : {dt_gscv.best_score_:.2f}')
```

In [ ]:
```
dt_best = DecisionTreeClassifier(criterion='entropy',max_depth=6)
dt_best.fit(X_train_predator, y_train_predator)
```

In [ ]:
```
importances = dt_best.feature_importances_
features_dict = dict(zip(predator_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=predator_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of Decision Tree")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.1.3 Random Forest

In [ ]:
```
# normal random forest
rfc = RandomForestClassifier(n_estimators=50, max_depth=5)
rfc.fit(X_train_predator,y_train_predator)
print(f"The train score is:",rfc.score(X_train_predator,y_train_predat
print(f"The test score is:",rfc.score(X_test_predator,y_test_predator)
```

In [ ]:
```
# cross validation
rfc_cv_scores = cross_val_score(rfc, X_train_predator, y_train_predato
rfc_cv_scores
```

In [ ]:
```
depths = [2,4,6,8,10]
train_scores,test_scores = validation_curve(RandomForestClassifier(n_e
                                            X_train_predator, y_train_
                                            param_name='max_depth', pa
mean_train_scores = np.average(train_scores, axis=1)
mean_test_scores = np.average(test_scores, axis=1)
pd.DataFrame([mean_train_scores.round(2),mean_test_scores.round(2)],
             columns=pd.Series(depths,name='max_depth'),
             index=['mean_train_scores','mean_test_scores'])
```

In [ ]:
```
# Grid Search
params = {'n_estimators':[10,50,100,150],'max_depth':[2,3,4,5,6]}
rfc_gscv = GridSearchCV(estimator=RandomForestClassifier(random_state=
rfc_gscv.fit(X_train_predator, y_train_predator)
print(f'rfc best hyperparams      : {rfc_gscv.best_params_}')
print(f'rfc best mean cv accuracy : {rfc_gscv.best_score_:.2f}')
```

In [ ]:
```
rfc_best = RandomForestClassifier(max_depth=6,n_estimators=10)
rfc_best.fit(X_train_predator, y_train_predator)
```

In [ ]:
```
importances = rfc_best.feature_importances_
features_dict = dict(zip(predator_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=predator_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of Random Forest")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.1.4 Adaboost

In [ ]:
```
ada = AdaBoostClassifier(n_estimators=50, learning_rate=1)
ada.fit(X_train_predator,y_train_predator)
print(f"The train score is:",ada.score(X_train_predator,y_train_predat
print(f"The test score is:",ada.score(X_test_predator,y_test_predator)
```

In [ ]:
```
# Grid Search
params = {'n_estimators':[10,50,100,150],'learning_rate':[0.01,0.1,0.5
ada_gscv = GridSearchCV(estimator=AdaBoostClassifier(random_state=123)
ada_gscv.fit(X_train_predator, y_train_predator)
print(f'ada best hyperparams      : {ada_gscv.best_params_}')
print(f'ada best mean cv accuracy : {ada_gscv.best_score_:.2f}')
```

In [ ]:
```
ada_best = AdaBoostClassifier(learning_rate=0.5,n_estimators=50)
ada_best.fit(X_train_predator, y_train_predator)
```

In [ ]:
```
importances = ada_best.feature_importances_
features_dict = dict(zip(predator_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=predator_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of AdaBoost")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.2 Classify Prey taxon

In [ ]:
```
ordinalencoder = OrdinalEncoder()
prey_x['Geographic location'] = ordinalencoder.fit_transform(prey_x[['
prey_x['Specific habitat'] = ordinalencoder.fit_transform(prey_x[['Spe
```

In [ ]:
```
from sklearn.model_selection import train_test_split

# Train Test Split
X_train_prey, X_test_prey, y_train_prey, y_test_prey = train_test_spli




# Scaling
scaler = StandardScaler()
X_train_prey = scaler.fit_transform(X_train_prey)
X_test_prey = scaler.transform(X_test_prey)
```

4.2.1 SVC

In [ ]:
```
svc = SVC(C=1,gamma='scale')
svc.fit(X_train_prey,y_train_prey)
print(f"The train score is:",svc.score(X_train_prey,y_train_prey))
print(f"The test score is:",svc.score(X_test_prey,y_test_prey))
```

In [ ]:
```
# Grid Search
params = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01]}
svc_gscv = GridSearchCV(estimator = SVC(random_state=123), param_grid=
svc_gscv.fit(X_train_prey, y_train_prey)
print(f'svc best hyperparams      : {svc_gscv.best_params_}')
print(f'svc best mean cv accuracy : {svc_gscv.best_score_:.2f}')
```

In [ ]:
```
svc_best = SVC(C=10,gamma=1,kernel='linear')
svc_best.fit(X_train_prey,y_train_prey)
```

In [ ]:
```
importances = svc_best.coef_[0]
features_dict = dict(zip(prey_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=prey_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of SVC")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.2.2 Decision Tree

In [ ]:
```
dt = DecisionTreeClassifier(criterion='gini', max_depth = 5)
dt.fit(X_train_prey,y_train_prey)
print(f"The train score is:",dt.score(X_train_prey,y_train_prey))
print(f"The test score is:",dt.score(X_test_prey,y_test_prey))
```

In [ ]:
```
# Grid Search
params = {'criterion': ['entropy'], 'max_depth' : [2,3,4,5,6]}
dt_gscv = GridSearchCV(estimator = DecisionTreeClassifier(random_state
dt_gscv.fit(X_train_prey, y_train_prey)
print(f'decision tree best hyperparams      : {dt_gscv.best_params_}')
print(f'decision tree best mean cv accuracy : {dt_gscv.best_score_:.2f
```

In [ ]:
```
dt_best = DecisionTreeClassifier(criterion='entropy',max_depth=6)
dt_best.fit(X_train_prey, y_train_prey)
```

In [ ]:
```
importances = dt_best.feature_importances_
features_dict = dict(zip(prey_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=prey_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of Decision Tree")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.2.3 Random Forest

In [ ]:
```
# normal random forest
rfc = RandomForestClassifier(n_estimators=50, max_depth=5)
rfc.fit(X_train_prey,y_train_prey)
print(f"The train score is:",rfc.score(X_train_prey,y_train_prey))
print(f"The test score is:",rfc.score(X_test_prey,y_test_prey))
```

In [ ]:
```
# cross validation
rfc_cv_scores = cross_val_score(rfc, X_train_prey, y_train_prey, cv=5,
rfc_cv_scores
```

In [ ]:
```
depths = [2,4,6,8,10]
train_scores,test_scores = validation_curve(RandomForestClassifier(n_e
                                            X_train_prey, y_train_prey
                                            param_name='max_depth', pa
mean_train_scores = np.average(train_scores, axis=1)
mean_test_scores = np.average(test_scores, axis=1)
pd.DataFrame([mean_train_scores.round(2),mean_test_scores.round(2)],
             columns=pd.Series(depths,name='max_depth'),
             index=['mean_train_scores','mean_test_scores'])
```

In [ ]:
```
# Grid Search
params = {'n_estimators':[10,50,100,150],'max_depth':[2,3,4,5,6]}
rfc_gscv = GridSearchCV(estimator=RandomForestClassifier(random_state=
rfc_gscv.fit(X_train_prey, y_train_prey)
print(f'random forest best hyperparams      : {rfc_gscv.best_params_}'
print(f'random forest best mean cv accuracy : {rfc_gscv.best_score_:.2
```

In [ ]:
```
rfc_best = RandomForestClassifier(max_depth=6,n_estimators=10)
rfc_best.fit(X_train_prey, y_train_prey)
```

In [ ]:
```
importances = rfc_best.feature_importances_
features_dict = dict(zip(prey_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=prey_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of Random Forest")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

4.2.4 Adaboost

In [ ]:
```
ada = AdaBoostClassifier(n_estimators=50, learning_rate=1)
ada.fit(X_train_prey,y_train_prey)
print(f"The train score is:",ada.score(X_train_prey,y_train_prey))
print(f"The test score is:",ada.score(X_test_prey,y_test_prey))
```

In [ ]:
```
# Grid Search
params = {'n_estimators':[10,50,100,150],'learning_rate':[0.01,0.1,0.5
ada_gscv = GridSearchCV(estimator=AdaBoostClassifier(random_state=123)
ada_gscv.fit(X_train_prey, y_train_prey)
print(f'ada best hyperparams      : {ada_gscv.best_params_}')
print(f'ada best mean cv accuracy : {ada_gscv.best_score_:.2f}')
```

In [ ]:
```
ada_best = AdaBoostClassifier(learning_rate=1,n_estimators=10)
ada_best.fit(X_train_prey, y_train_prey)
```

In [ ]:
```
importances = ada_best.feature_importances_
features_dict = dict(zip(prey_x.columns, abs(importances)))
important_features = sorted(features_dict.items(), key=lambda x: -x[1]
important_features
```

In [ ]:
```
sns.barplot(x=prey_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of AdaBoost")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```

5 PCA

5.1 PCA for predator

In [ ]:
```
pca = PCA(n_components=3, random_state=123)

X_train_pca_predator = pca.fit_transform(X_train_predator)
X_test_pca_predator = pca.transform(X_test_predator)

pca.explained_variance_ratio_
```

In [ ]:
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X_train_pca_predator[:,0],X_train_pca_predator[:,1],X_train
plt.title('PCA for Predator')
```

In [ ]:
```
fig, ax=plt.subplots(nrows=1, ncols=3, figsize=(15,5))

sns.scatterplot(x = X_train_pca_predator[:,0], y = X_train_pca_predato
ax[0].set_title("Dimensions 0&1 of the PCA Transformation")

sns.scatterplot(x = X_train_pca_predator[:,0], y = X_train_pca_predato
ax[1].set_title("Dimensions 0&2 of the PCA Transformation")

sns.scatterplot(x = X_train_pca_predator[:,1], y = X_train_pca_predato
ax[2].set_title("Dimensions 1&2 of the PCA Transformation")
```

5.2 PCA for prey

In [ ]:
```
pca = PCA(n_components=3, random_state=123)

X_train_pca_prey = pca.fit_transform(X_train_prey)
X_test_pca_prey = pca.transform(X_test_prey)

pca.explained_variance_ratio_
```

In [ ]:
```
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X_train_pca_prey[:,0],X_train_pca_prey[:,1],X_train_pca_pre
plt.title('PCA for Predator')
```

In [ ]:
```
fig, ax=plt.subplots(nrows=1, ncols=3, figsize=(15,5))

sns.scatterplot(x = X_train_pca_prey[:,0], y = X_train_pca_prey[:,1],
ax[0].set_title("Dimensions 0&1 of the PCA Transformation")

sns.scatterplot(x = X_train_pca_prey[:,0], y = X_train_pca_prey[:,2],
ax[1].set_title("Dimensions 0&2 of the PCA Transformation")

sns.scatterplot(x = X_train_pca_prey[:,1], y = X_train_pca_prey[:,2],
ax[2].set_title("Dimensions 1&2 of the PCA Transformation")
```

6 Clustering Model: Kmeans

6.1 Kmeans to cluster predator

In [ ]:
```
inertia = []
for i in range(1, 11):
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(predator_x)
    inertia.append(km.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
```

In [ ]:
```
kmeans_predator = KMeans(n_clusters=2, random_state=123)
kmeans_predator.fit(predator_x)
labels_predator = kmeans_predator.labels_
labels_predator = pd.DataFrame(labels_predator, columns=["kmeans_label
labels_predator.value_counts()
```

In [ ]:
```
X_predator_cluster = pd.concat([predator_x, predator_y, labels_predato
X_predator_cluster.groupby('Predator common name').mean()
```

In [ ]:
```
y_train_predator.value_counts()
```

6.2 Kmeans to cluster prey

In [ ]:
```
inertia = []
for i in range(1, 11):
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(prey_x)
    inertia.append(km.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
```

In [ ]:
```
kmeans_prey = KMeans(n_clusters=2, random_state=123)
kmeans_prey.fit(prey_x)
labels_prey = kmeans_prey.labels_
labels_prey = pd.DataFrame(labels_prey, columns=["kmeans_label"])
labels_prey.value_counts()
```

In [ ]:
```
X_prey_cluster = pd.concat([prey_x, prey_y, labels_prey], axis=1)
X_prey_cluster.groupby('Prey taxon').mean()
```

7. Regression Model

In [ ]:
```
merged_df = pd.merge(df_predator, df_prey, left_index=True, right_inde

# print(merged_df.columns)

merged_df['Mass difference'] = merged_df['Predator mass'] - merged_df[
# merged_df['Mass difference'] = merged_df['Predator length'] - merged

reg_x = merged_df[['Type of feeding interaction', 'Geographic location
                   'Latitude Minute_x', 'Latitude label_x', 'Longitude
                   'Longitude label_x', 'Depth_x', 'Mean annual temp_x
                   'Mean PP_x', 'SD PP_x', 'Specific habitat_x']]

reg_y = merged_df['Mass difference']
```

In [ ]:
```
reg_y
```

In [ ]:
```
non_numeric_columns = reg_x.select_dtypes(exclude=['int64', 'float64']

ordinalencoder = OrdinalEncoder()

for col_name in list(non_numeric_columns):
#     print(col_name)
    reg_x[col_name] = reg_x[col_name].astype(str)
    reg_x[col_name] = ordinalencoder.fit_transform(reg_x[[col_name]]).
```

In [ ]:
```
# Train Test Split
X_train_len, X_test_len, y_train_len, y_test_len = train_test_split(re
                                                                    re
                                                                    te
                                                                    ra


# Scaling
scaler = StandardScaler()
X_train_len = scaler.fit_transform(X_train_len)
X_test_len = scaler.transform(X_test_len)
```

7.1 Linear Regression

In [ ]:
```
reg = LinearRegression()
reg.fit(X_train_len,y_train_len)
print(f"The train score:", reg.score(X_train_len,y_train_len))
print(f"The test score:", reg.score(X_test_len,y_test_len))
```

7.2 Ridge Regression

In [ ]:
```
ridge = Ridge(alpha=1)
ridge.fit(X_train_len,y_train_len)
print(f"The train score:", ridge.score(X_train_len,y_train_len))
print(f"The test score:", ridge.score(X_test_len,y_test_len))
```

In [ ]:
```
params = {'alpha':[0.01,0.1,0.5,1]}
ridge_gscv = GridSearchCV(estimator=Ridge(), param_grid=params, cv=3,
ridge_gscv.fit(X_train_len, y_train_len)
print(f'ada best hyperparams      : {ridge_gscv.best_params_}')
print(f'ada best mean cv accuracy : {ridge_gscv.best_score_:.2f}')
```

7.3 Lasso Regression

In [ ]:
```
lasso = Lasso(alpha=1)
lasso.fit(X_train_len,y_train_len)
print(f"The train score:", lasso.score(X_train_len,y_train_len))
print(f"The test score:", lasso.score(X_test_len,y_test_len))
```

In [ ]:
```
params = {'alpha':[0.01,0.1,0.5,1]}
lasso_gscv = GridSearchCV(estimator=Lasso(), param_grid=params, cv=3,
lasso_gscv.fit(X_train_len, y_train_len)
print(f'ada best hyperparams      : {lasso_gscv.best_params_}')
print(f'ada best mean cv accuracy : {lasso_gscv.best_score_:.2f}')
```

7.5 Decision Tree Regression

In [ ]:
```
dtr = DecisionTreeRegressor(max_depth=5)
dtr.fit(X_train_len,y_train_len)
print(f"The train score:", dtr.score(X_train_len,y_train_len))
print(f"The test score:", dtr.score(X_test_len,y_test_len))
```

In [ ]:
```
params = {'criterion': ['mse', 'mae'], 'max_depth' : [2,3,4,5,6]}
dt_gscv = GridSearchCV(estimator = DecisionTreeRegressor(random_state=
dt_gscv.fit(X_train_len, y_train_len)
print(f'decision tree best hyperparams      : {dt_gscv.best_params_}')
print(f'decision tree best mean cv accuracy : {dt_gscv.best_score_:.2f
```

In [ ]:
```
df = pd.DataFrame(dtr.feature_importances_, index=reg_x.columns).sort_
df['index'] = df['index'].str.split('_').str[0]
df
```

In [ ]:
```
sns.set_style("whitegrid")
sns.set(rc={'figure.figsize':(11.7,5)})

# Create the bar chart using seaborn's barplot function
ax = sns.barplot(y=0, x='index', data=df)

# Set the title and axis labels
ax.set_title("Feature Importance for Decision Tree")
ax.set_xlabel("Importance")
ax.set_ylabel("Feature")

# Show the plot
plt.show()
```

5.6 Adaboost Regression

In [ ]:
```
adarg = AdaBoostRegressor(n_estimators=100, random_state=123)
adarg.fit(X_train_len,y_train_len)
print(f"The train score:", adarg.score(X_train_len,y_train_len))
print(f"The test score:", adarg.score(X_test_len,y_test_len))
```

In [ ]:
```
params = {'n_estimators':[10,50,100,150],'learning_rate':[0.01,0.1,0.5
ada_gscv = GridSearchCV(estimator=AdaBoostRegressor(random_state=123),
ada_gscv.fit(X_train_len, y_train_len)
print(f'ada best hyperparams      : {ada_gscv.best_params_}')
print(f'ada best mean cv accuracy : {ada_gscv.best_score_:.2f}')
```

In [ ]:
```
df = pd.DataFrame(adarg.feature_importances_, index=reg_x.columns).sor
df['index'] = df['index'].str.split('_').str[0]
df
```

In [ ]:
```
sns.set_style("whitegrid")
sns.set(rc={'figure.figsize':(11.7,5)})

# Create the bar chart using seaborn's barplot function
ax = sns.barplot(y=0, x='index', data=df)

# Set the title and axis labels
ax.set_title("Feature Importance for AdaBoost")
ax.set_xlabel("Importance")
ax.set_ylabel("Feature")

# Show the plot
plt.show()
```

5.6 Random Forest Regression

In [ ]:
```
rf = RandomForestRegressor(max_depth=5)
rf.fit(X_train_len,y_train_len)
print(f"The train score:", dtr.score(X_train_len,y_train_len))
print(f"The test score:", dtr.score(X_test_len,y_test_len))


depths = [2,4,6,8,10]
train_scores,test_scores = validation_curve(RandomForestRegressor(n_es
                                            X_train_len, y_train_len,
                                            param_name='max_depth', pa
mean_train_scores = np.average(train_scores, axis=1)
mean_test_scores = np.average(test_scores, axis=1)
pd.DataFrame([mean_train_scores.round(2),mean_test_scores.round(2)],
             columns=pd.Series(depths,name='max_depth'),
             index=['mean_train_scores','mean_test_scores'])
```

In [ ]:
```
params = {'n_estimators':[10,50,100,150],'max_depth':[2,3,4,5,6]}
rfc_gscv = GridSearchCV(estimator=RandomForestRegressor(random_state=1
rfc_gscv.fit(X_train_len, y_train_len)
print(f'random forest best hyperparams      : {rfc_gscv.best_params_}'
print(f'random forest best mean cv accuracy : {rfc_gscv.best_score_:.2
```

In [ ]:
```
df = pd.DataFrame(rf.feature_importances_, index=reg_x.columns).sort_v
df['index'] = df['index'].str.split('_').str[0]
df
```

In [ ]:
```
sns.set_style("whitegrid")
sns.set(rc={'figure.figsize':(11.7,5)})

# Create the bar chart using seaborn's barplot function
ax = sns.barplot(y=0, x='index', data=df)

# Set the title and axis labels
ax.set_title("Feature Importance for Random Forest")
ax.set_xlabel("Importance")
ax.set_ylabel("Feature")

# Show the plot
plt.show()
```