Project introduction

Link to Notebook

- Link to notebook in repository

** Please use github gist link to see the interactive visualizations in the **Visualization** section. In the github viewer, these visualizations are not shown properly.

This is the Group Project of DataX.

Problem: With the expansion of humans, many animals have been harmed having to suffer through habitat loss, poaching, and more. Over time, many animals species are becoming endangered and some have gone extinct. Over a thousand animal species are endangered, and in the past few centuries a few hundred animal species have gone extinct. In an attempt to mitigate greater damage to the animal population and biodiversity, our group DataX will try to find out what specific factors causes animal endangerment as well as find locations where animal endangerment is more likely to occur.

Hypothesis: Specific factors that cause animal endangerment will be caused or related to human activities

Data Sources:

Here Several files filed with mammal data or read, cleaned, and joined together. For this project we decided to focus on the mammals as a subset of animals so it would be easier to work with our data

- Links to find these datasets:
    1. (https://www.iucnredlist.org/resources/spatial-data-download)
    2. (https://www.iucnredlist.org/search?taxonomies=100041&searchType=species)
    3. (https://zenodo.org/records/6644198)
- Link to DBF opener(https://www.dbfopener.com/)
- Link to another DBF opener(https://www.vertopal.com/en/convert/dbf-to-csv)

The some of these csv files are origionally dbf files. We convert them to csv files using the two sites provided above. These are also all sources that give information related to animals and there endangerment/status

Any Changes

We explored our aggregated data from different sources. We added some spatial data about the country of each species to our previous dataset.

```
import pandas as pd
import numpy as np
import unittest
import io, time, json
import requests
%matplotlib inline
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import os
from getpass import getpass
import seaborn as sns
from matplotlib.pyplot import figure
# import .py files
import dataCleaning
import mergeMammalInformation
import mammalTrends
import endagermentOverTime
import threatsToSpecies
import endangermentLevels
import geoDataFile
import featureManipulation
import warnings
warnings.filterwarnings("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive/CollabNotebooks')
```

Import Data

Either use Github or Google Drive to import the datasets.

Github

```
# Clone github repository to access cleaned datasets
git_user = getpass("Your github user: ")
git_token = getpass("Your github pass / token: ")
os.environ["GITHUB_USER"] = git_user
os.environ["GITHUB_TOKEN"] = git_token
```

```
!git clone https://"$GITHUB_USER":"$GITHUB_TOKEN"@github.com/uic-ds-fall-2023/endangered-species-analysis-datax.git
%cd ./endangered-species-analysis-datax
```

Google Drive

```
# colab-specific file access
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/My Drive/Colab Notebooks
```

```
Mounted at /content/drive
/content/drive/My Drive/Colab Notebooks
```

Acquiring & Cleaning

This is where our data is brought in and cleaned. Below you will find csv files read in different code boxes and several operations such as `drop()` or `drop_duplicates()` performed to get rid of any unneeded data to reduce redundancy.

There are merges between datasets that happen in this section. The first 5 boxes below are data obtained from (https://www.iucnredlist.org/resources/spatial-data-download) which has information on different types of mammal species and is broken up into:

- MAMMALS_FRESHWATER.csv
- MAMMALS_MARINE.csv
- MAMMALS_MARINE_AND_TERRESTRIAL.csv
- MAMMALS_TERRESTRIAL_ONLY.csv

This is MAMMALS_FRESHWATER, MAMMALS_MARINE_ONLY, MAMMALS_MARINE_AND_TERRESTRIAL, and MAMMALS_TERRESTRIAL_ONLY dataset and cleaning. All 4 are merged into one dataframe called Mammals

```
## Call function from dataCleaning.py to clean and merge datasets
Mammals = dataCleaning.cleanMammalData()
#Mammals.head()
```

This is infomation from IUCN on all mammals obtainted through their search list. These files were also added as part of our dataset because they provide some more information on mammal species that we need such as threats and habitat

1. Go to this link: https://www.iucnredlist.org/search?taxonomies=100041&searchType=species
2. click on download in the right corner
3. click search results
4. go to you account page to complete the download warning: this may not get the search results you want. You must edit you account profile to include download options you are interested in such as habitats and threats

This information is split into 4 different files and are meant to be joined together using assessmentId and internalTaxonId.

These files include:

- assessments.csv :3 columns of redlistCriteria, criteriaVersion, language were taken out because they were not needed. Rows where redlistCategory is least concern, data deficient, and not evaluated are taken out as well bcause these species are not going to be facing threats
- habitats.csv: here the code column is removed because we don't have a use for it
- taxonomy.csv: here many columns are removed because we don't have a use for them and because some columns were completely filled with NaN values
- threats.csv: At the end all dataframes are merged into a dataframe called table3

```
assessments = pd.read_csv('assessments.csv', sep=',')
habitats = pd.read_csv('habitats.csv', sep=',')
taxonomy = pd.read_csv('taxonomy.csv', sep=',')
threats = pd.read_csv('threats.csv', sep=',')
assessments, habitats, taxonomy, threats, table3= mergeMammalInformation.mergeMammalInfo(assessments, habitats, taxonomy, threats
```

MDD_species.csv is the last csv file used for this project. This one is brought in because it provides spatial data of species which we can use later on.

```
Species = pd.read_csv("MDD_species.csv")
```


Data Analysis

Our 'Mammals' data includes various details about mammals such as their scientific name, presence, origin, if they are currently extinct, type of mammal, Genus, and various other details that aid with identification. We will use these details to correctly label, group, and analyze mammals when considering causes and prevention of mammal endangerment

```
grouped = Mammals.groupby('SEASONAL')

# Define a function to check all the 'LEGEND' values in each group
def check_legend(group):
    # Use the .unique() method to get unique 'LEGEND' values in the group
    unique_legends = group['LEGEND'].unique()
    return unique_legends

# Apply the custom function to each group and print the results
result = grouped.apply(check_legend)
# Display the result
#result
```

2. Table 3

As Stated Prior Table3 is the combination of: assessments.csv, habitats.csv, taxonomy.csv, threats.csv. All of these csv where joined using either the internalTaxonId or the assessmentId. In addition the CSV's have been cleaned to get rid of any duplicate reports and or null values and any kind of report in the threat.csv that does not indicate an endagered species.

As a result we have information such as names and scientific names of the species, habitats, threats and their severity, last date that a species was seen and whether they are possibly extinct in the wild, etc.

```
#table3.head()
```

```
#endagermentOverTime.mammalEndangerment(mammalTrends, Mammals, assessments) ## call function from endangermentOverTime.py to plo
```

Utilizing the datasets and uncommenting the code above, we can observe mammal endegerment over the year 2008-2022 based on the type of mammal that is classified.

We used the year published and as our X-axis in this graph and we use the classification of the type of mammal for the Y-axis.

X: year Y: level of endagerment

Visualization

Here we are working on the visualizations of the project. First step is figuring out what data we want to use to create our visualization. Many of the boxes below look through the existing clean dataframes and pick apart the pieces that are used for the visualizations. A lot of work also goes into getting the longitute and latitude values and processing them into something usable for our visualization. We import/download python packages and a library called folium to help with map visualization.

Load previously cleaned data, plus a dataset for geo spatial data of species "MDD_species.csv".

```
# Get cleaned data
data, mammals, species, mammals_marine, mammals_terrestrial, mammals_freshwater = dataCleaning.cleanDataframes(table3, Mammals,
#mammals.head()
assessment_result = list(data["redlistCategory"].unique())
#assessment_result
```

Add necessary python packages for map visualization.

```
!pip -q install pycountry-convert
!pip -q install geopy
!pip -q install folium
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.1/10.1 MB 51.9 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 227.5/227.5 kB 19.3 MB/s eta 0:00:00
  Building wheel for pycountry (pyproject.toml) ... done
```

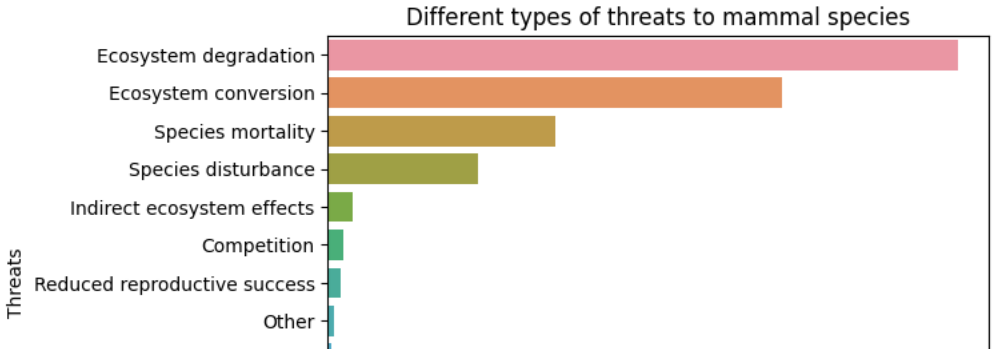
```
from pycountry_convert import country_alpha2_to_continent_code, country_name_to_country_alpha2

def get_country(col):
    try:
        cn_a2_code =  country_name_to_country_alpha2(col)
    except:
        cn_a2_code = 'Unknown'
    try:
        cn_continent = country_alpha2_to_continent_code(cn_a2_code)
    except:
        cn_continent = 'Unknown'
    return (cn_a2_code, cn_continent)
countries = species["countryDistribution"].apply(lambda con: get_country(con))
species["country"] = countries
#species.head()
```

```
geo_data = geoDataFile.getGeoData(data, mammals, species)  ## call function in geoDataFile to retrieve geo data
```

```
assessment_result = [
    'Near Threatened',
    'Vulnerable',
    'Endangered',
    'Critically Endangered',
    'Extinct in the Wild',
    'Extinct'
]
```

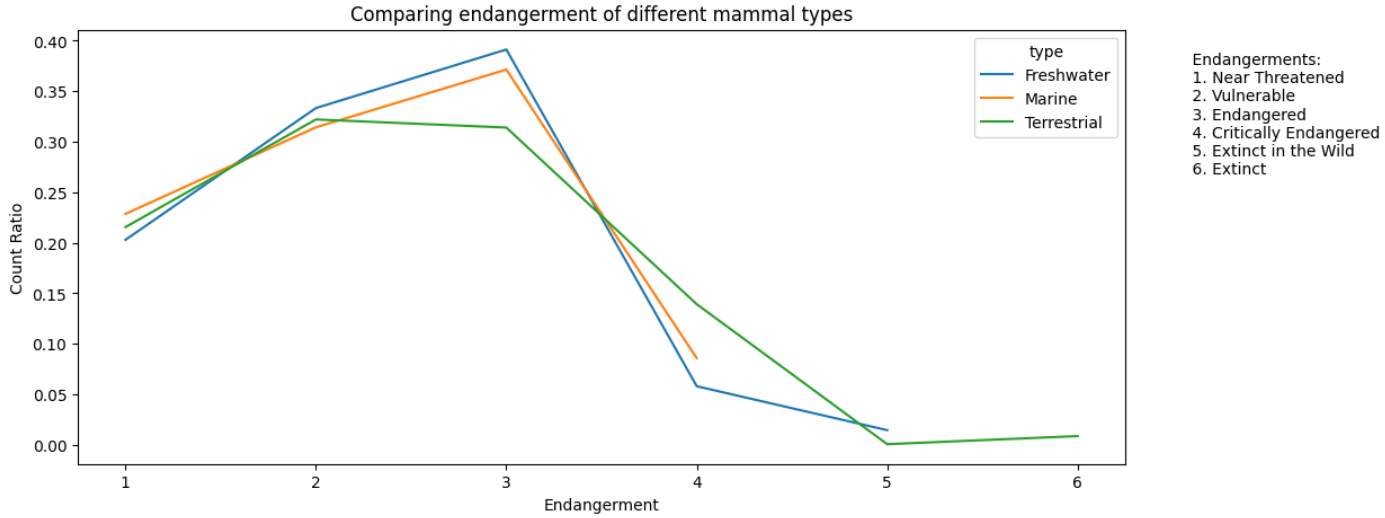
More Visualization!

```
table3, Mammals, Species, threats = threatsToSpecies.plotThreatsToSpecies(table3, Mammals, Species, threats)  ## call function i
```



As we can see from the above plot, the most important threat reason is related to humans. Human acitivities can easily increase the Ecosystem Degradation and as a result threaten mammal species.

```
data, mammals = endangermentLevels.plotCompareEndangerment(data, mammals) # call function in endangermentLevels.py in order to p
```



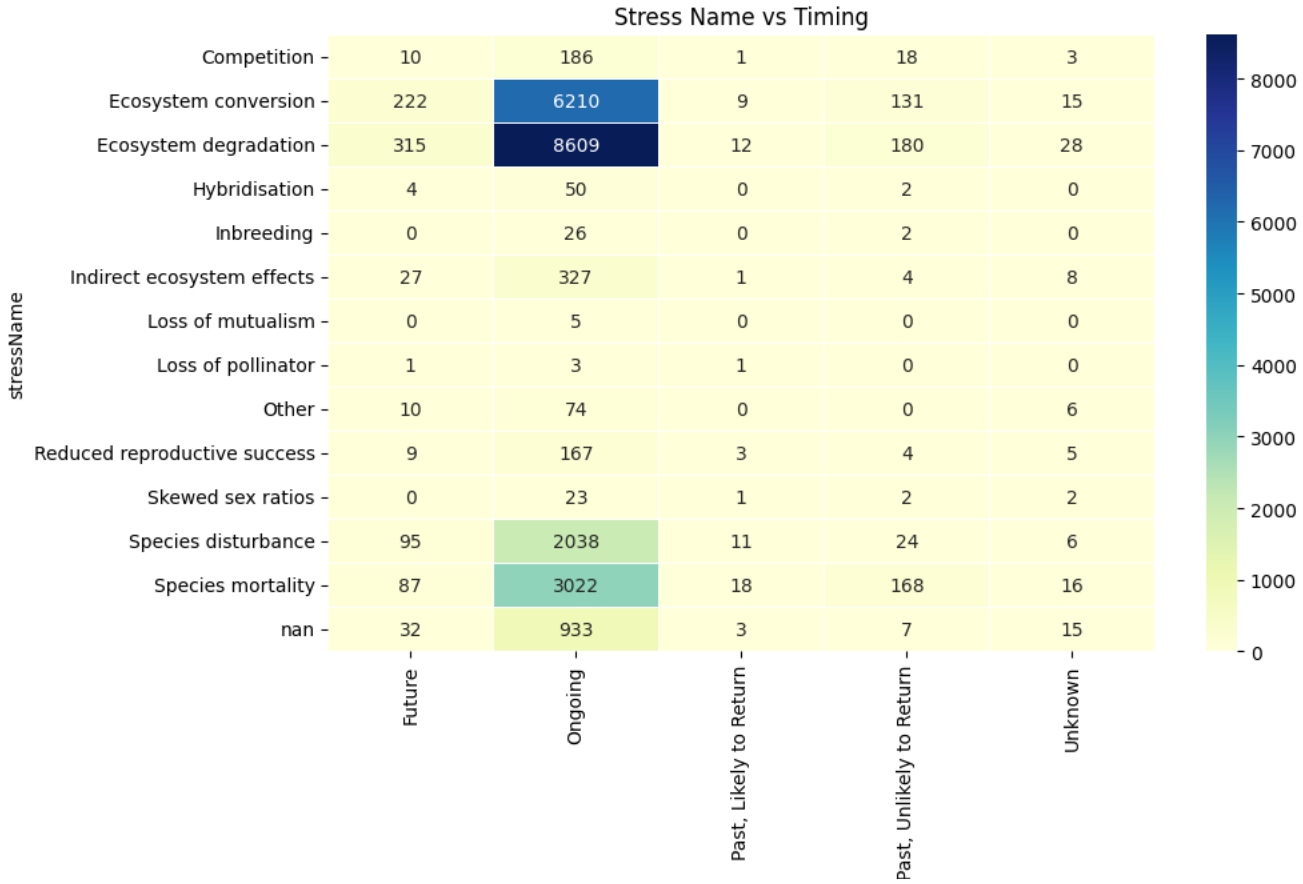
As we can see from the above plot, the Terrestrial mammals have more endangered if we pick top 3 classes ("Crtitically endangered", "Extinct in wild", "Exinct") as ciritical ones. If we only check the label "Endangered", freshwater mammals are more rather than the Marine mammals.

Additional Visualization

In this visualization we observe the threat name to mammals and whether they are predicted to occur in the future, are ongoing, they occured in the past and are likely to return or unlikely to occur and the unkown threats.

As we can see from our recorded data that the ongoing threats are still big in number but there is quite a few predited to still be an issue in the future. There are some cases that are unlikely to occur but these either are due to extinction or something was done to remove the threat to the mammal entirely.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.DataFrame(threats)
heatmap_data = df.pivot_table(index='stressName', columns='timing', aggfunc='size', fill_value=0)
plt.figure(figsize=(10,6))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt='g', linewidths=.5)
plt.title('Stress Name vs Timing')
plt.show()
```



Machine Learning Analysis

Predicting Endangerment

In this part we want to predict the level of endangerment of speices given their features in our dataset. The features are based on physical charecteristic of species, geo spatial data and simple data about the family of speices.

```
data = table3
mammals = Mammals
species = Species
# More Cleaning!
species["sciName"] = species["sciName"].apply(lambda x: str(x).lower().replace("_", " "))
data["scientificName_x"] = data["scientificName_x"].apply(lambda x: str(x).lower())
mammals["SCI_NAME"] = mammals["SCI_NAME"].apply(lambda x: str(x).lower())
data["target"] = data["redlistCategory"].apply(lambda x: 0 if x in ["Vulnerable", "Near Threathened"] else 1)
data["target"].value_counts()
```

```
1    25656
0    14606
Name: target, dtype: int64
```

```
geo_data = geo_data[["scientificName_x", "countryDistribution"]]
```

```
data = data[["target", "redlistCategory", "scientificName_x", "systems", "realm"]]
mammals = mammals[["SCI_NAME", "MARINE", "TERRESTIAL", "FRESHWATER", "FAMILY", "SHAPE_LENG", "SHAPE_AREA"]]
merged = data.merge(mammals, left_on="scientificName_x", right_on="SCI_NAME")
merged = merged.merge(geo_data, left_on="scientificName_x", right_on="scientificName_x")
merged = merged.drop_duplicates(ignore_index=True)
#merged.head()
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

le = LabelEncoder()
merged["realm"] = le.fit_transform(merged["realm"])
merged["FAMILY"] = le.fit_transform(merged["FAMILY"])
merged["countryDistribution"] = le.fit_transform(merged["countryDistribution"])
merged["MARINE"] = merged["MARINE"].apply(lambda x: 1 if (x or x == "true" or x =="True") else 0)
merged["TERRESTIAL"] = merged["TERRESTIAL"].apply(lambda x: 1 if (x or x == "true" or x =="True") else 0)
merged["FRESHWATER"] = merged["FRESHWATER"].apply(lambda x: 1 if (x or x == "true" or x =="True") else 0)

X = merged[["realm", "MARINE", "TERRESTIAL", "FRESHWATER", "FAMILY", "SHAPE_LENG", "SHAPE_AREA", "countryDistribution"]]
y = merged["target"]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

We use two models, decision tree and a simple neural network with two hidden layers, to predict the endangerment of species given their features. The decision tree model gives us a better accuracy on the training set. We will plot 4 layers of this decision tree model.

```
X_train.shape, X_test.shape
```

```
((3816, 8), (1272, 8))
```

```
test_accuracies = []
train_accuracies = []
```

Neural Network

We used a neural network with different hidden layers as the hyper parameters. The accuracy of the neural network is not so high, because these are complicated models that require a larger amount of data to be trained on. As a result, we get low accuracy on both train and test dataset for neural network model.

```
from sklearn.neural_network import MLPClassifier

model_mlp_3 = MLPClassifier(random_state=0, hidden_layer_sizes=(10, 15, 10))
model_mlp_3.fit(X_train, y_train)
test_accuracies.append(model_mlp_3.score(X_test, y_test))
train_accuracies.append(model_mlp_3.score(X_train, y_train))
print("Train Acc: ", model_mlp_3.score(X_train, y_train))
print("Test Acc: ", model_mlp_3.score(X_test, y_test))
```

```
Train Acc:  0.6776729559748428
Test Acc:  0.6839622641509434
```

```
model_mlp_4 = MLPClassifier(random_state=0, hidden_layer_sizes=(10, 20, 20, 10))
model_mlp_4.fit(X_train, y_train)
test_accuracies.append(model_mlp_4.score(X_test, y_test))
train_accuracies.append(model_mlp_4.score(X_train, y_train))
print("Train Acc: ", model_mlp_4.score(X_train, y_train))
print("Test Acc: ", model_mlp_4.score(X_test, y_test))
```

```
Train Acc:  0.6949685534591195
Test Acc:  0.6988993710691824
```

Decision Tree

The second model is decision tree. This a simple model that is trained well on our data. It gives us an accuracy of 0.99 on train and 0.88 on test dataset.

```
from sklearn.tree import DecisionTreeClassifier

model_dt = DecisionTreeClassifier(random_state=0)
model_dt.fit(X_train, y_train)
test_accuracies.append(model_dt.score(X_test, y_test))
train_accuracies.append(model_dt.score(X_train, y_train))
print("Train Acc: ", model_dt.score(X_train, y_train))
print("Test Acc: ", model_dt.score(X_test, y_test))
```

```
Train Acc:  0.9992138364779874
Test Acc:  0.8883647798742138
```

Random Forest

Combining multiple simple decision tree models into a random forest model also gives us a better performance. We used 10, 20, and 30 estimators. By increasing the number of estimators, the accuracy of the model on both train and test datasets increased.

```
from sklearn.ensemble import RandomForestClassifier

model_rf_10 = RandomForestClassifier(n_estimators=10, random_state=1)
model_rf_10.fit(X_train, y_train)
test_accuracies.append(model_rf_10.score(X_test, y_test))
train_accuracies.append(model_rf_10.score(X_train, y_train))
print("model_rf 10 on Train: ", model_rf_10.score(X_train, y_train))
print("model_rf 10 on Test: ", model_rf_10.score(X_test, y_test))

model_rf_20 = RandomForestClassifier(n_estimators=20, random_state=1)
model_rf_20.fit(X_train, y_train)
test_accuracies.append(model_rf_20.score(X_test, y_test))
train_accuracies.append(model_rf_20.score(X_train, y_train))
print("model_rf 20 on Train: ", model_rf_20.score(X_train, y_train))
print("model_rf 20 on Test: ", model_rf_20.score(X_test, y_test))

model_rf_30 = RandomForestClassifier(n_estimators=30, random_state=1)
model_rf_30.fit(X_train, y_train)
test_accuracies.append(model_rf_30.score(X_test, y_test))
train_accuracies.append(model_rf_30.score(X_train, y_train))
print("model_rf 30 on Train: ", model_rf_30.score(X_train, y_train))
print("model_rf 30 on Test: ", model_rf_30.score(X_test, y_test))
```

```
model_rf 10 on Train:  0.9942348008385744
model_rf 10 on Test:  0.8781446540880503
model_rf 20 on Train:  0.9973794549266247
model_rf 20 on Test:  0.8922955974842768
model_rf 30 on Train:  0.9986897274633124
model_rf 30 on Test:  0.9025157232704403
```

Interpreting the Models

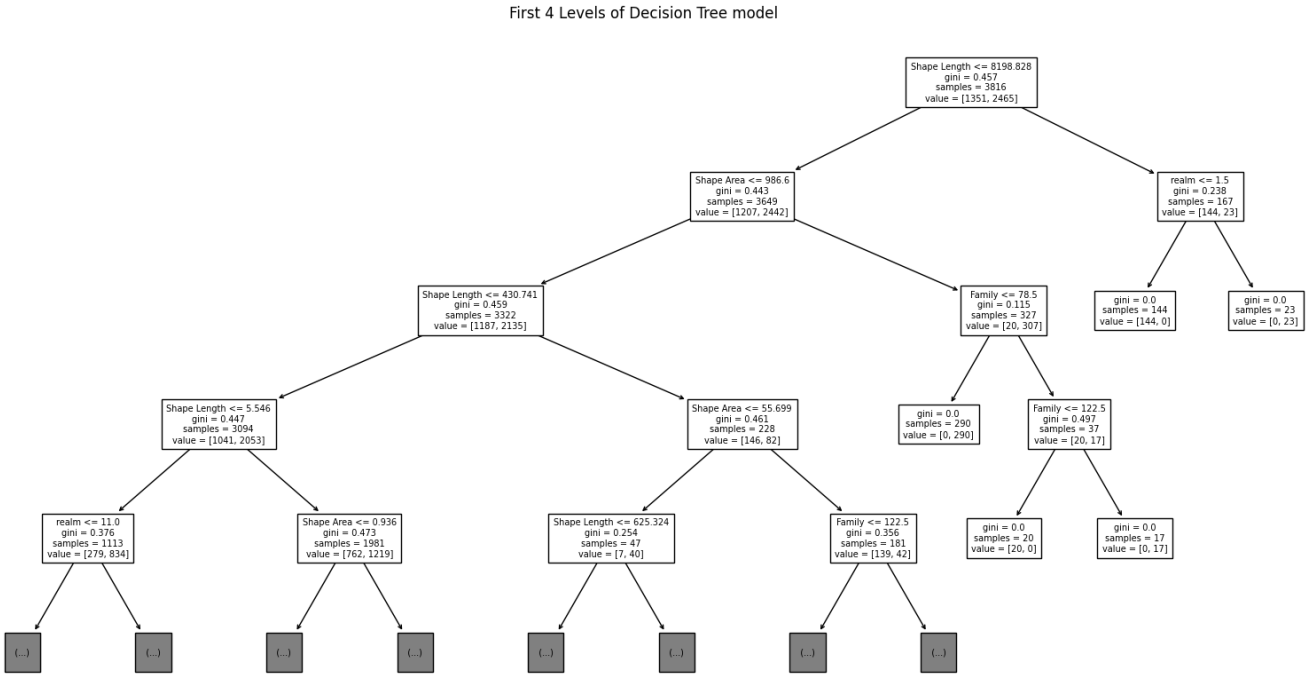
> Reference: https://towardsdatascience.com/best-practice-to-calculate-and-interpret-model-feature-importance-14f0e11ee660

Here we try to interpret our ML models to see which features are more important for them to drive decisions. This tells us based on different features of species, which of them might be more improtant in their endagerment.

For a decision tree, we plotted the first 4 levels of the tree. As we can see the features about physical characteristic of species are more important for this model to make decisions.

```
from sklearn import tree

plt.figure(figsize=(20, 10))
tree.plot_tree(model_dt, max_depth=4, fontsize=7, feature_names=["realm", "Marine", "Terrestrial", "Freshwater", "Family", "Shap
plt.title("First 4 Levels of Decision Tree model")
```

```
Text(0.5, 1.0, 'First 4 Levels of Decision Tree model')
```

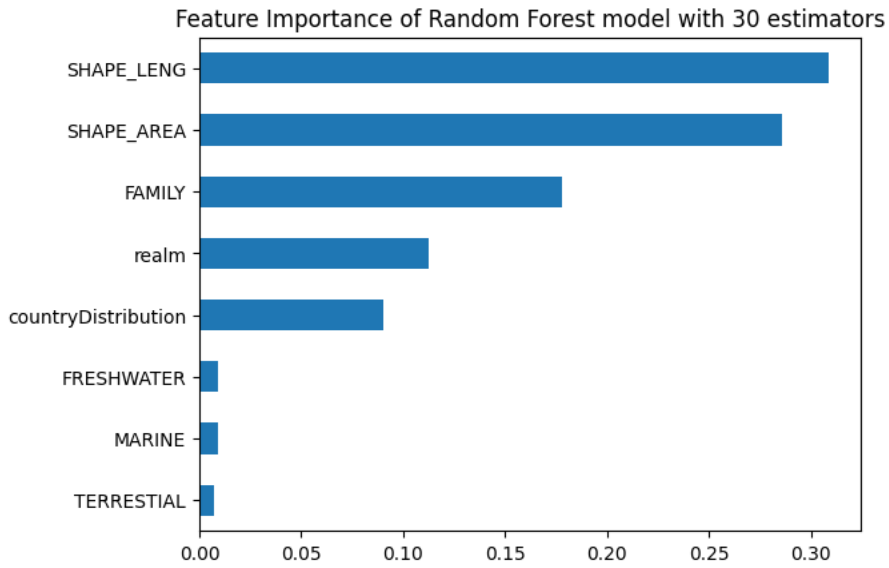


The is a method to detect important features of an ML model called "Drop Column". During this methodd, we drop a column of the dataset and train the model on new reduced dataset and see the difference on the model's accuracy compared to baseline.

The more drop on the accuracy means that the feature is more important for the model to draw decisions.

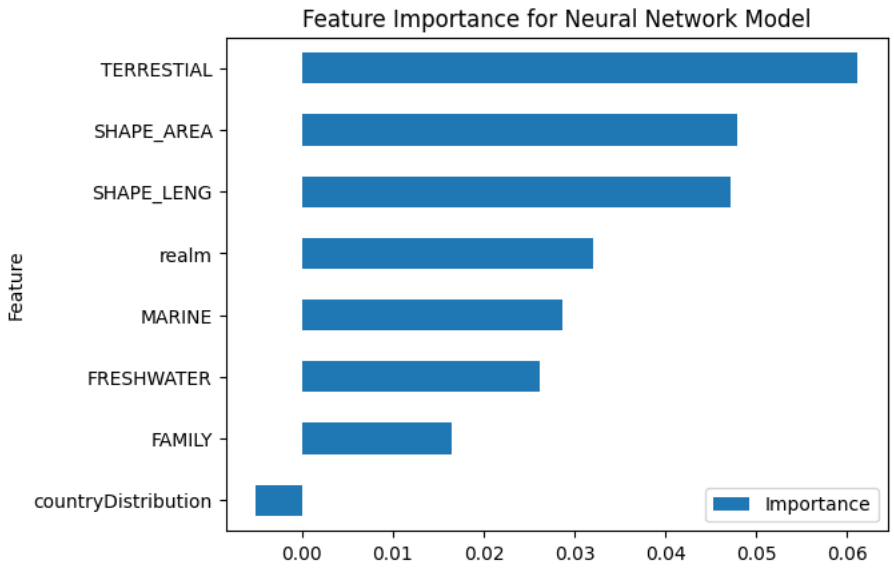
```
from matplotlib.pyplot import figure
feat_importances = pd.Series(model_rf_30.feature_importances_, index = X_train.columns).sort_values(ascending = True)
feat_importances.plot(kind = 'barh')
plt.title("Feature Importance of Random Forest model with 30 estimators")
```

```
Text(0.5, 1.0, 'Feature Importance of Random Forest model with 30 estimators')
```



As we can see, for both Neural Network and Random Forest model, the features about physical characteristic (SHAPE) of species are more important. Also the features about family of the model and their bio-geographycal location is among the important features for neural network model

```
featureManipulation.plotImportances(model_mlp_4, X, y) ## call function in featureManipulation.py in order to plot the Feature im
```



Results

Overall we used 3 different sources to make 3 dataframes which we used as our dataset. Through data analysis we saw how animal endangerment has changed over time in terms of conservation status. Our first 3 visualization focused on using locational data of all mmammal species in our dataset and creating different types of species distributions. These visualizations allowed us to look at how mammal species were scattered around the world in terms of numbers, concentration, and conservation status/threat level. This really helped give an idea of where too look by seeing which locations hadd the greatest number of threatened species assoicated with them. We found that there are mor animal species towards the center of contienents and that the number of threatened species also follows this trend. Next was a bar graph on what types of threats the mammal species in our dataset were facing and found the biggest threat is ecosystem degredation followed by ecosystem conversion. our next visualization looks at different types of mammals(terrestial, marine, freshwater) and looks the ratio of them that have a certain conservation status. From this we see that there is a higher ratio of terrestial species with worse conservation status such as extinct and extinct in the wild. Marine mammals have the lowest ratio of with worse conservation status. This visualization points towards the conclusion that marine species are better off because they have the least contact with humans but we can't say that without more evidence. Our last visualization is a heap map looking at threats a species faces and the likely hood they will face them in the future or whether they faced them in the past. This shows what may species are facing threats right now and which once are the most intense. Also gives some idea of how species faced threats in a chronological point of view. In the ML section we look at different models from Neural Network, Decesion tree, and Random forrest models with different estmaiteors and we managed to acheive high accuracy with some of these models. Out of all used models the Random Forest model with 30 estimators provides best results with Train accuracy of 99% and test accuracy of 90%. The most important feature when making such predictions is Shape_Leng followed by Shape_Area