In [180]:
```
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 [2]:
```
# Load Data
df = pd.read_csv("5241dataset.csv", encoding = 'unicode_escape')
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1370
471252.py:2: DtypeWarning: Columns (27,41) have mixed types. Specify
dtype option on import or set low_memory=False.
  df = pd.read_csv("5241dataset.csv", encoding = 'unicode_escape')
```

1. Dataset Description

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34931 entries, 0 to 34930
Data columns (total 49 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Record number                          34931 non-null  int64

 1   Individual ID                          33200 non-null  float64
 2   Predator                               34931 non-null  object
 3   Predator common name                   34931 non-null  object
 4   Predator taxon                         34931 non-null  object
 5   Predator lifestage                     34931 non-null  object
 6   Type of feeding interaction            34931 non-null  object
 7   Predator length                        34931 non-null  float64
 8   Predator length unit                   34931 non-null  object
 9   Predator standard length               34930 non-null  float64
 10  Predator total length                  34818 non-null  float64
 11  Predator TL/FL/SL conversion reference 29683 non-null  object
 12  Standardised predator length           34931 non-null  float64
 13  Predator mass                          34931 non-null  float64
 14  Predator mass unit                     34931 non-null  object
 15  Predator mass check                    34931 non-null  float64
 16  Predator mass check diff               34931 non-null  float64
 17  Predator ratio mass/mass               34931 non-null  float64
 18  SI predator mass                       34931 non-null  float64
 19  Diet coverage                          34931 non-null  object
 20  Prey                                   34931 non-null  object
 21  Prey common name                       34931 non-null  object
 22  Prey taxon                             34931 non-null  object
 23  Prey length                            34931 non-null  float64
 24  Prey length unit                       34929 non-null  object
 25  SI prey length                         34931 non-null  float64
 26  Prey width                             964 non-null    float64
 27  Prey width unit                        964 non-null    object
 28  Prey mass                              34931 non-null  float64
 29  Prey mass unit                         34931 non-null  object
 30  Prey mass check                        34931 non-null  float64
 31  Prey mass check diff                   34931 non-null  float64
 32  Prey ratio mass/mass                   34931 non-null  float64
 33  SI prey mass                           34931 non-null  float64
 34  Geographic location                    34931 non-null  object
 35  Latitude                               34931 non-null  object
 36  Latitude Degree                        34931 non-null  int64
 37  Latitude Minute                        34931 non-null  int64
 38  Latitude label                         34931 non-null  object
 39  Longitude                              34931 non-null  object
 40  Longitude Degree                       34931 non-null  int64
 41  Longitude Minute                       34931 non-null  object
 42  Longitude label                        34931 non-null  object
 43  Depth                                  34931 non-null  int64
 44  Mean annual temp                       34931 non-null  float64
 45  SD annual temp                         34931 non-null  float64
 46  Mean PP                                34931 non-null  int64
 47  SD PP                                  34931 non-null  float64
 48  Specific habitat                       34931 non-null  object
dtypes: float64(21), int64(6), object(22)
memory usage: 13.1+ MB
```

In [4]:
```
df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%
Record number	34931.0	17466.000000	10083.855463	1.000000e+00	8733.50000	17466.0000
Individual ID	33200.0	7767.209699	5603.773065	1.000000e+00	2066.75000	7000.0000
Predator length	34931.0	411.708758	865.149724	3.000000e+00	67.50000	200.0000
Predator standard length	34930.0	275.498247	851.527130	0.000000e+00	18.50000	92.0000
Predator total length	34818.0	410.125251	891.934132	0.000000e+00	73.41000	210.0000
Standardised predator length	34931.0	68.362248	53.988482	3.000000e-01	28.92000	68.0000
Predator mass	34931.0	15292.363560	46260.898001	1.140000e-04	172.01000	1751.1000
Predator mass check	34931.0	57433.601425	142121.384771	1.413000e-03	1265.80000	16455.0000
Predator mass check diff	34931.0	42141.261535	97206.742764	1.298700e-03	995.64000	12997.0000
Predator ratio mass/mass	34931.0	7.488458	5.037282	1.053500e+00	4.11160	6.5149
SI predator mass	34931.0	15292.363560	46260.898001	1.140000e-04	172.01000	1751.1000
Prey length	34931.0	72.617237	214.978864	8.000000e-02	4.68225	21.9890
SI prey length	34931.0	8.128151	9.084534	4.160000e-04	2.19945	5.0000
Prey width	964.0	266.169233	238.438741	4.530000e-01	110.00000	216.0000
Prey mass	34931.0	28.409896	121.990812	7.530000e-11	0.16240	1.4888
Prey mass check	34931.0	264.665719	2501.661458	3.770000e-12	0.55684	6.5417
Prey mass check diff	34931.0	236.258016	2460.368288	-1.151000e+02	0.22081	4.6817
Prey ratio mass/mass	34931.0	8.610662	35.534542	1.915600e-02	3.19750	5.2333
SI prey mass	34931.0	28.407359	121.991367	7.530000e-11	0.16240	1.4888
Latitude Degree	34931.0	40.118605	12.041278	8.000000e+00	40.00000	40.0000
Latitude Minute	34931.0	9.778821	21.456202	0.000000e+00	0.00000	0.0000
Longitude Degree	34931.0	47.276545	28.059336	2.000000e+00	14.00000	70.0000
Depth	34931.0	1561.806418	1768.576051	5.000000e+00	677.00000	677.0000
Mean annual temp	34931.0	15.166128	6.466898	-1.300000e+00	13.90000	15.1000
SD annual temp	34931.0	4.366302	2.285000	6.000000e-01	2.50000	5.6000
Mean PP	34931.0	727.152243	320.066435	9.100000e+01	437.00000	867.0000
SD PP	34931.0	69.225888	34.382417	7.000000e+00	38.00000	80.0000

2. Data Cleaning

2.1 Missing Values

In [5]:
```python
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()
```

```
The columns contain missing values are:
 Index(['Individual ID', 'Predator standard length', 'Predator total
length',
       'Predator TL/FL/SL conversion reference', 'Prey length unit',
       'Prey width', 'Prey width unit'],
      dtype='object')
```

In [6]:
```python
df.drop(['Individual ID','Predator TL/FL/SL conversion reference','Pre
```

In [7]:
```python
df.columns
```

Out[7]:
```
Index(['Record number', 'Predator', 'Predator common name', 'Predator
taxon',
       'Predator lifestage', 'Type of feeding interaction', 'Predator
length',
       'Predator length unit', 'Predator standard length',
       'Predator total length', 'Standardised predator length',
       'Predator mass', 'Predator mass unit', 'Predator mass check',
       'Predator mass check diff', 'Predator ratio mass/mass',
       'SI predator mass', 'Diet coverage', 'Prey', 'Prey common name
',
       'Prey taxon', 'Prey length', 'Prey length unit', 'SI prey leng
th',
       'Prey mass', 'Prey mass unit', 'Prey mass check',
       'Prey mass check diff', 'Prey ratio mass/mass', 'SI prey mass'
,
       'Geographic location', 'Latitude', 'Latitude Degree', 'Latitud
e Minute',
       'Latitude label', 'Longitude', 'Longitude Degree', 'Longitude
Minute',
       'Longitude label', 'Depth', 'Mean annual temp', 'SD annual tem
p',
       'Mean PP', 'SD PP', 'Specific habitat'],
      dtype='object')
```

2.2 Dataset Split

In [8]:
```python
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 [9]:
```
df_predator['Predator length unit'].value_counts()
```

Out[9]:
```
mm    22069
cm    12423
µm      439
Name: Predator length unit, dtype: int64
```

In [10]:
```
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
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/5563
51888.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_predator['Predator length'][i] = df_predator['Predator length'][
i]*0.1
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/5563
51888.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_predator['Predator length'][i] = df_predator['Predator length'][
i]*0.0001
```

In [11]:
```
df_predator['Predator mass unit'].value_counts()
```

Out[11]:
```
g    34931
Name: Predator mass unit, dtype: int64
```

In [12]:
```
df_predator.drop(['Predator length unit'], axis = 1, inplace = True)
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/2564
476634.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_predator.drop(['Predator length unit'], axis = 1, inplace = True
)
```

In [13]:
```
df_predator['Predator common name'].value_counts()
```

Out[13]:
```
Albacore                   3581
Spurdog / spiny dogfish    3287
Atlantic cod               2518
Yellowfin tuna             2113
Atlantic bluefin tuna      1909
                           ...
Tadpole sculpin               2
4-spot megrim                 1
tope                          1
Imperial scaldfish            1
Red Irish lord                1
Name: Predator common name, Length: 87, dtype: int64
```

In [14]:
```
major_predator = df_predator['Predator common name'].value_counts()/df
major_predator_names = major_predator[major_predator == True].index
major_predator_names
```

Out[14]:
```
Index(['Albacore', 'Spurdog / spiny dogfish', 'Atlantic cod', 'Yellow
fin tuna',
       'Atlantic bluefin tuna', 'Bigeye tuna'],
      dtype='object')
```

In [15]:
```
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 [16]:
```
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 [17]:
```
df_prey['Prey length unit'].value_counts()
```

Out[17]:
```
mm    22471
cm    11489
µm      969
Name: Prey length unit, dtype: int64
```

In [18]:
```
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
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/9469
70684.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_prey['Prey length'][i] = df_prey['Prey length'][i]*0.1
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/9469
70684.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_prey['Prey length'][i] = df_prey['Prey length'][i]*0.0001
```

In [19]:
```
df_prey['Prey mass unit'].value_counts()
```

Out[19]:
```
g     34728
mg      203
Name: Prey mass unit, dtype: int64
```

In [20]:
```
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
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1295
962586.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_prey['Prey mass'][i] = df_prey['Prey mass'][i]*0.001
```

In [21]:
```
df_prey.drop(['Prey length unit','Prey mass unit'], axis = 1, inplace
```

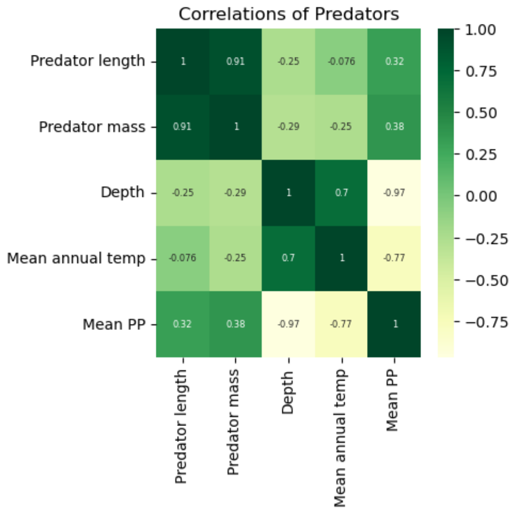
```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/3688
790300.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_prey.drop(['Prey length unit','Prey mass unit'], axis = 1, inpla
ce = True)
```

In [22]:
```
prey_y = df_prey['Prey taxon']
prey_x = df_prey[['Prey length','Prey mass', 'Geographic location','De
```

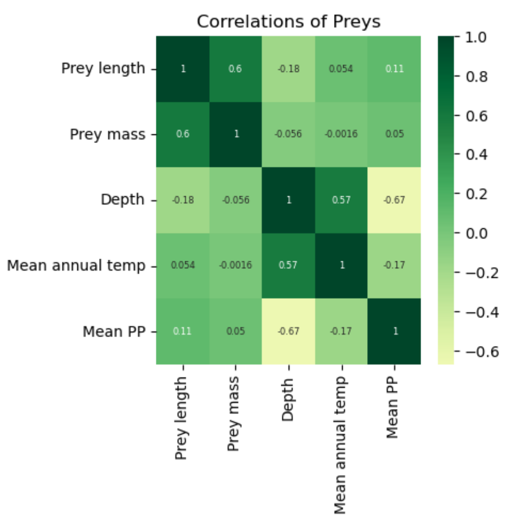
3. Data Visualization

3.1 Correlations of Independent Variables

In [23]:
```
# 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 [24]:
```
# 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()
```

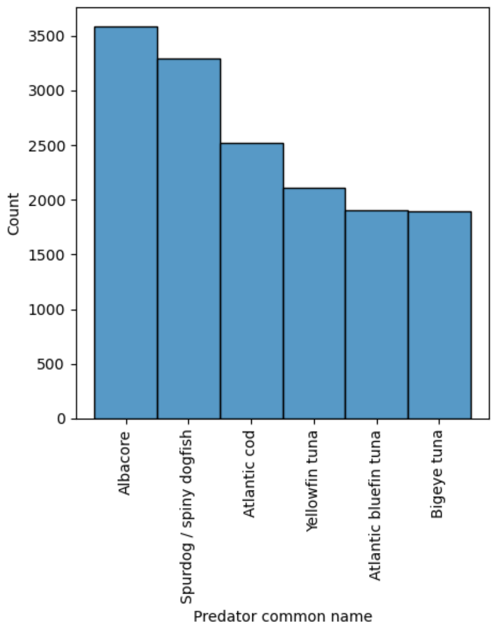


In [25]:
```
df['Prey taxon'].unique()
```

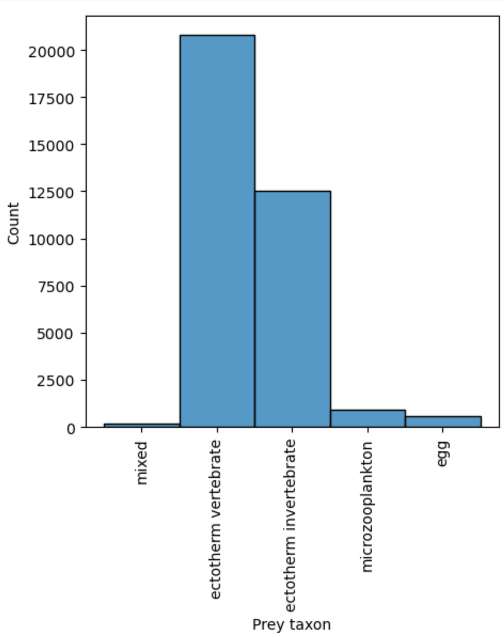
Out[25]:
```
array(['mixed', 'ectotherm vertebrate', 'ectotherm invertebrate',
       'microzooplankton', 'egg'], dtype=object)
```

3.2 Distribution of Dependent Variable

In [26]:
```
fig, ax=plt.subplots(figsize=(5,5))
sns.histplot(predator_y, ax=ax)
plt.xticks(rotation = 90)
plt.show()
```



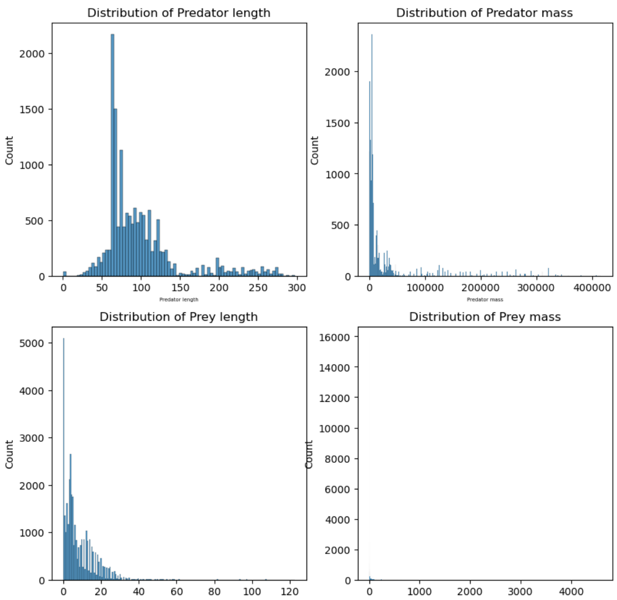
In [27]:
```
fig, ax=plt.subplots(figsize=(5,5))
sns.histplot(prey_y, ax=ax)
plt.xticks(rotation = 90)
plt.show()
```



3.3 Numerial Features

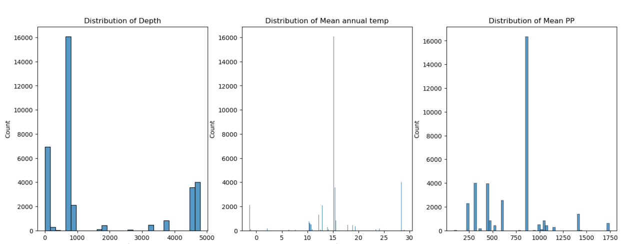
In [68]:
```
#predator and prey
df_v = pd.concat([predator_x,prey_x],axis=1)

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()
```

4



In [76]:
```
# 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()
```

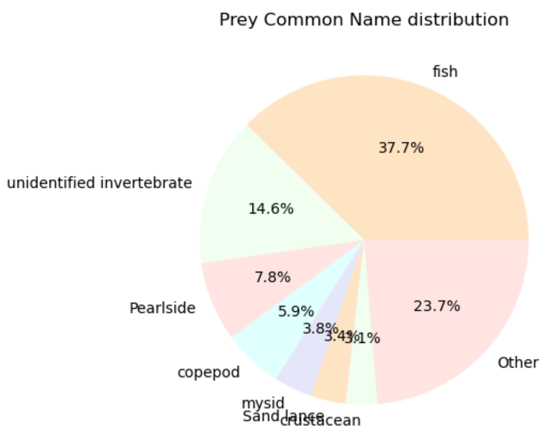
3



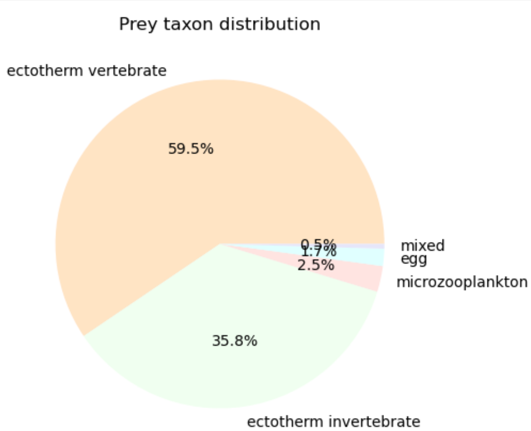
3.4 Categorical Features

In [77]:
```
# 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()
```

```
Index(['fish', 'unidentified invertebrate', 'Pearlside', 'copepod', '
mysid',
       'Sand lance', 'crustacean'],
      dtype='object')
```



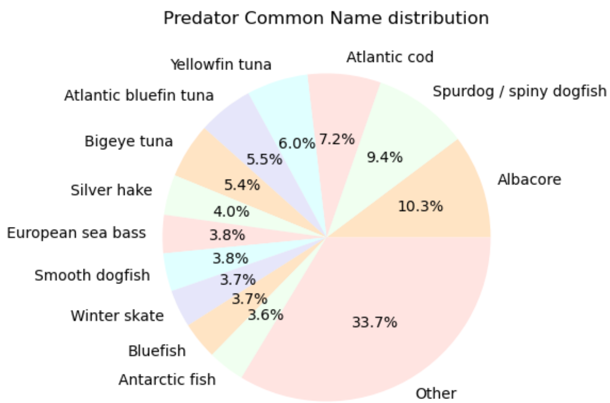
In [78]:
```
# Prey taxon
def myformat(value):
    return f'{value:.1f}%'

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()
```



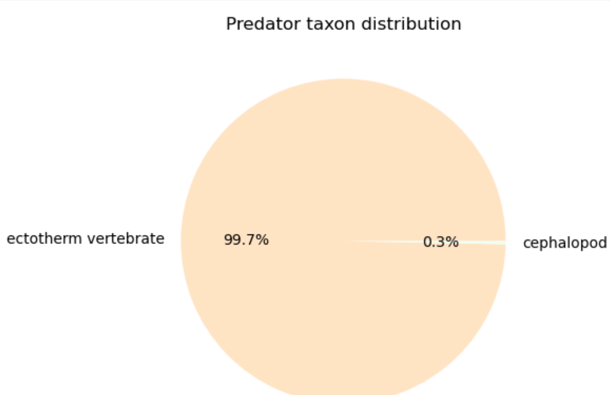
In [79]:
```
# predator common names
def myformat(value):
    return f'{value:.1f}%'

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()
```

```
Index(['Albacore', 'Spurdog / spiny dogfish', 'Atlantic cod', 'Yellow
fin tuna',
       'Atlantic bluefin tuna', 'Bigeye tuna', 'Silver hake',
       'European sea bass', 'Smooth dogfish', 'Winter skate', 'Bluefi
sh',
       'Antarctic fish'],
      dtype='object')
```



In [80]:
```
# 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 [81]:
```
# 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
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1166
39362.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  predator_x['Predator taxon'] = ordinalencoder.fit_transform(predato
r_x[['Predator taxon']]).reshape(1,-1).tolist()[0]
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1166
39362.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  predator_x['Predator lifestage'] = ordinalencoder.fit_transform(pre
dator_x[['Predator lifestage']]).reshape(1,-1).tolist()[0]
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1166
39362.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  predator_x['Type of feeding interaction'] = ordinalencoder.fit_tran
sform(predator_x[['Type of feeding interaction']]).reshape(1,-1).toli
st()[0]
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1166
39362.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  predator_x['Diet coverage'] = ordinalencoder.fit_transform(predator
_x[['Diet coverage']]).reshape(1,-1).tolist()[0]
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1166
39362.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead


See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  predator_x['Geographic location'] = ordinalencoder.fit_transform(pr
edator_x[['Geographic location']]).reshape(1,-1).tolist()[0]
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1166
39362.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  predator_x['Specific habitat'] = ordinalencoder.fit_transform(preda
tor_x[['Specific habitat']]).reshape(1,-1).tolist()[0]
```

In [82]:

```
# 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 [83]:

```
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)
```

```
The train score is: 0.893172165958837
The test score is: 0.8919007184846506
```

In [84]:

```
# 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}')
```

```
svc best hyperparams      : {'C': 10, 'gamma': 1}
svc best mean cv accuracy : 0.98
```

In [85]:
```
svc_best = SVC(C=10,gamma=1,kernel='linear')
svc_best.fit(X_train_predator,y_train_predator)
```

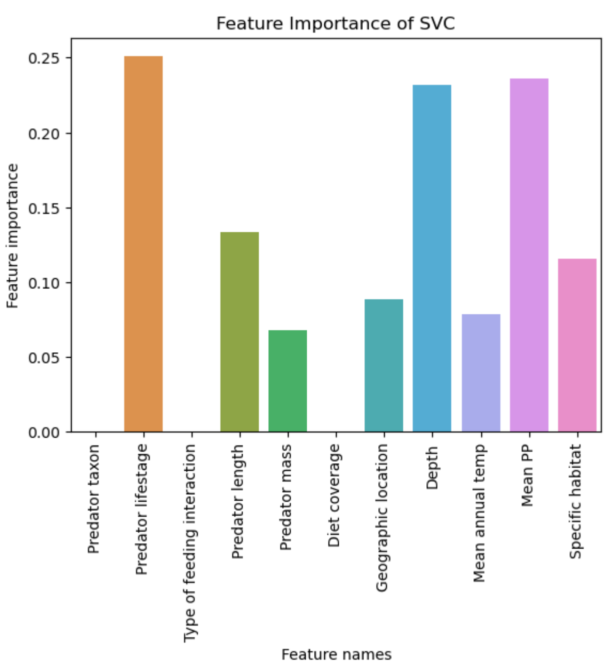
Out[85]:
```
SVC(C=10, gamma=1, kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]:
```
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
```

Out[86]:
```
[('Predator lifestage', 0.2507300387024649),
 ('Mean PP', 0.2360341213344168),
 ('Depth', 0.23191475135497236),
 ('Predator length', 0.13364966017144195),
 ('Specific habitat', 0.11552890874742702),
 ('Geographic location', 0.08890184213673946),
 ('Mean annual temp', 0.07868244720355236),
 ('Predator mass', 0.06816983848531796),
 ('Type of feeding interaction', 0.00043798373543962876),
 ('Predator taxon', 0.0),
 ('Diet coverage', 0.0)]
```

In [87]:
```
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 [88]:
```
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))
```

```
The train score is: 0.8951323097027115
The test score is: 0.8909209666884389
```

In [89]:
```
# 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}')
```

```
dt best hyperparams      : {'criterion': 'entropy', 'max_depth': 6}
dt best mean cv accuracy : 0.95
```

In [90]:
```
dt_best = DecisionTreeClassifier(criterion='entropy',max_depth=6)
dt_best.fit(X_train_predator, y_train_predator)
```

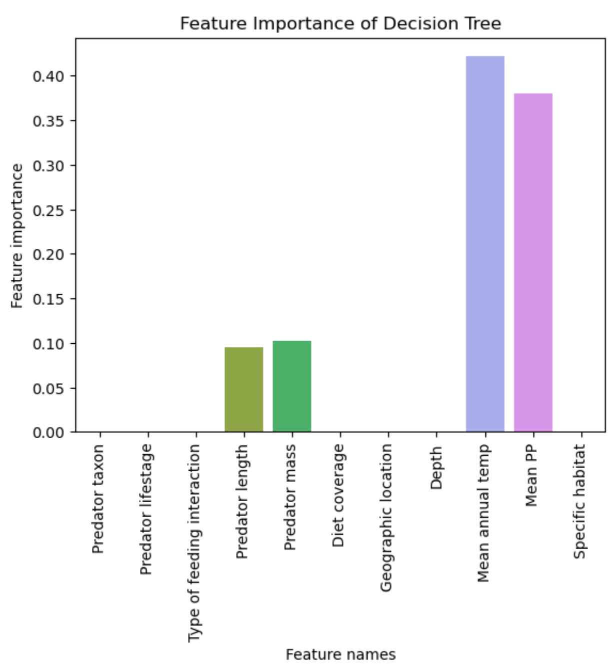
Out[90]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=6)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [91]:
```
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
```

Out[91]:
```
[('Mean annual temp', 0.4216168393869405),
 ('Mean PP', 0.38040805188666665),
 ('Predator mass', 0.10237907590256595),
 ('Predator length', 0.09559603282382692),
 ('Predator taxon', 0.0),
 ('Predator lifestage', 0.0),
 ('Type of feeding interaction', 0.0),
 ('Diet coverage', 0.0),
 ('Geographic location', 0.0),
 ('Depth', 0.0),
 ('Specific habitat', 0.0)]
```

In [92]:
```
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 [93]:
```
# 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)
```

```
The train score is: 0.9027278667102254
The test score is: 0.9036577400391901
```

In [94]:
```
# cross validation
rfc_cv_scores = cross_val_score(rfc, X_train_predator, y_train_predato
rfc_cv_scores
```

Out[94]:
```
array([0.89383422, 0.89914251, 0.89750919, 0.89873418, 0.90155229])
```

In [95]:
```
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'])
```

Out[95]:

max_depth	2	4	6	8	10
mean_train_scores	0.72	0.87	0.91	0.97	1.00
mean_test_scores	0.72	0.87	0.91	0.96	0.99

In [96]:
```
# 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}')
```

```
rfc best hyperparams      : {'max_depth': 6, 'n_estimators': 10}
rfc best mean cv accuracy : 0.92
```

In [110]:
```
rfc_best = RandomForestClassifier(max_depth=6,n_estimators=10)
rfc_best.fit(X_train_predator, y_train_predator)
```

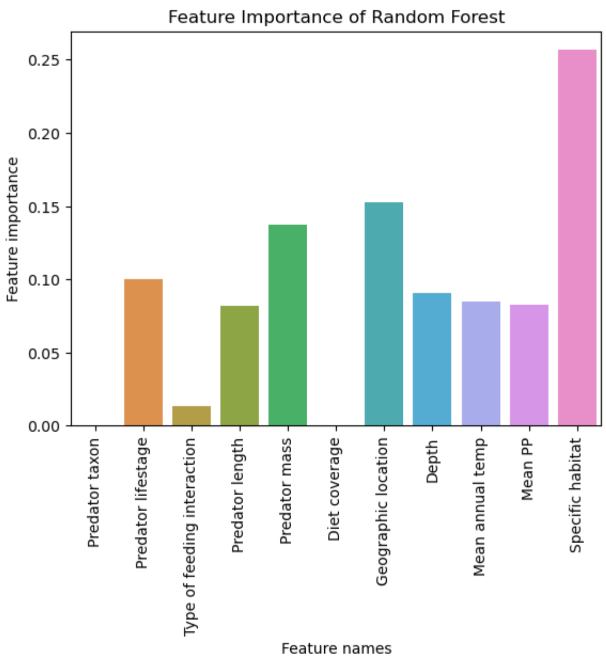
Out[110]:
```
RandomForestClassifier(max_depth=6, n_estimators=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [111]:
```
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
```

Out[111]:
```
[('Specific habitat', 0.256560492030104),
 ('Geographic location', 0.15289079064402605),
 ('Predator mass', 0.13723529462744416),
 ('Predator lifestage', 0.09976287851650764),
 ('Depth', 0.09067220494493555),
 ('Mean annual temp', 0.08455929899678413),
 ('Mean PP', 0.08292953223382424),
 ('Predator length', 0.08200977517086039),
 ('Type of feeding interaction', 0.013111345938932368),
 ('Diet coverage', 0.0002683868965814759),
 ('Predator taxon', 0.0)]
```

In [112]:
```
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 [113]:
```
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)
```

```
The train score is: 0.4780300555374061
The test score is: 0.48171129980404964
```

In [114]:
```
# 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}')
```

```
ada best hyperparams      : {'learning_rate': 0.5, 'n_estimators': 50
}
ada best mean cv accuracy : 0.83
```

In [115]:
```
ada_best = AdaBoostClassifier(learning_rate=0.5,n_estimators=50)
ada_best.fit(X_train_predator, y_train_predator)
```

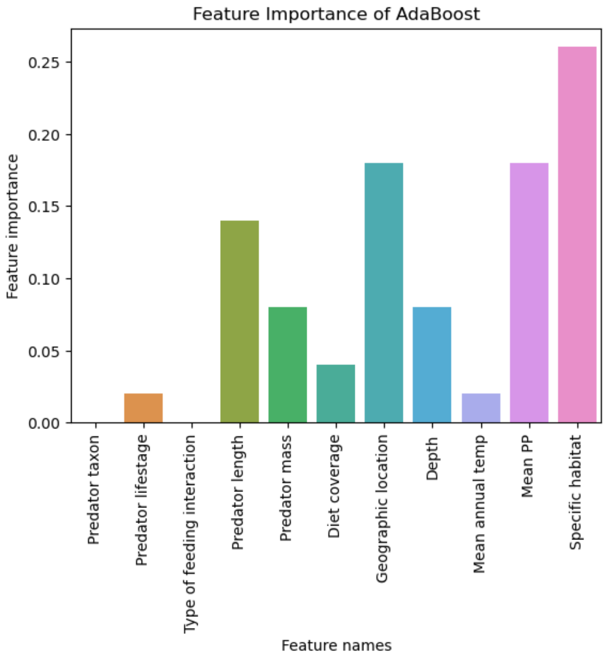
Out[115]:
```
AdaBoostClassifier(learning_rate=0.5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [116]:
```
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
```

Out[116]:
```
[('Specific habitat', 0.26),
 ('Geographic location', 0.18),
 ('Mean PP', 0.18),
 ('Predator length', 0.14),
 ('Predator mass', 0.08),
 ('Depth', 0.08),
 ('Diet coverage', 0.04),
 ('Predator lifestage', 0.02),
 ('Mean annual temp', 0.02),
 ('Predator taxon', 0.0),
 ('Type of feeding interaction', 0.0)]
```

In [117]:
```python
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 [118]:
```python
ordinalencoder = OrdinalEncoder()
prey_x['Geographic location'] = ordinalencoder.fit_transform(prey_x[['
prey_x['Specific habitat'] = ordinalencoder.fit_transform(prey_x[['Spe
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/3909
440097.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  prey_x['Geographic location'] = ordinalencoder.fit_transform(prey_x
[['Geographic location']]).reshape(1,-1).tolist()[0]
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/3909
440097.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  prey_x['Specific habitat'] = ordinalencoder.fit_transform(prey_x[['
Specific habitat']]).reshape(1,-1).tolist()[0]
```

In [119]:
```python
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 [120]:
```python
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))
```

```
The train score is: 0.8626896650443745
The test score is: 0.8605982539001001
```

In [121]:
```
# 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}')
```

```
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/joblib/externals/loky/process_executor.py:702: UserWarning: A w
orker stopped while some jobs were given to the executor. This can be
caused by a too short worker timeout or by a memory leak.
  warnings.warn(

svc best hyperparams      : {'C': 10, 'gamma': 1}
svc best mean cv accuracy : 0.93
```

In [122]:
```
svc_best = SVC(C=10,gamma=1,kernel='linear')
svc_best.fit(X_train_prey,y_train_prey)
```

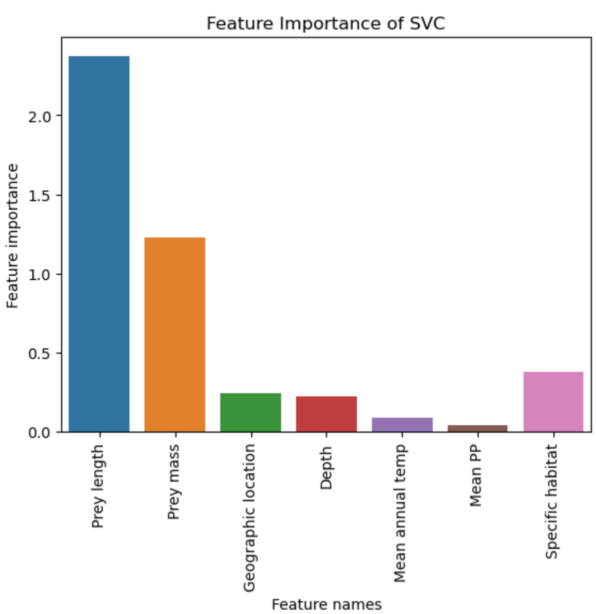
Out[122]:
```
SVC(C=10, gamma=1, kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [123]:
```
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
```

Out[123]:
```
[('Prey length', 2.37413915950674),
 ('Prey mass', 1.230711912501647),
 ('Specific habitat', 0.3780779543212702),
 ('Geographic location', 0.24032763379182143),
 ('Depth', 0.22540354848024435),
 ('Mean annual temp', 0.08728371815595892),
 ('Mean PP', 0.04359612810458202)]
```

In [124]:
```
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 [125]:
```
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))
```

```
The train score is: 0.8502004008016032
The test score is: 0.8512952626305997
```

In [133]:
```
# 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
```

```
decision tree best hyperparams      : {'criterion': 'entropy', 'max_d
epth': 6}
decision tree best mean cv accuracy : 0.90
```

In [134]:
```
dt_best = DecisionTreeClassifier(criterion='entropy',max_depth=6)
dt_best.fit(X_train_prey, y_train_prey)
```

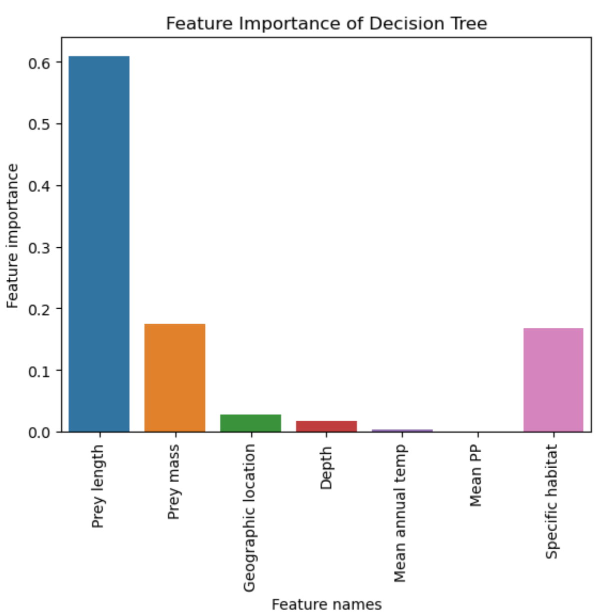
Out[134]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=6)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [135]:
```
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
```

Out[135]:
```
[('Prey length', 0.609117289870658),
 ('Prey mass', 0.17496708325855898),
 ('Specific habitat', 0.16818707327772858),
 ('Geographic location', 0.027699601547847008),
 ('Depth', 0.017340218522859562),
 ('Mean annual temp', 0.0026863035722660858),
 ('Mean PP', 2.4299500816036974e-06)]
```

In [136]:
```
sns.barplot(x=prey_x.columns, y=abs(importances))
plt.xticks(rotation=90)
plt.title("Feature Importance of Decision Tree")
plt.ylabel("Feature importance")
plt.xlabel("Feature names")
plt.ylabel("Feature importance")
plt.show()
```



4.2.3 Random Forest

In [137]:
```
# 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))
```

```
The train score is: 0.84740910392213
The test score is: 0.8494346643766996
```

In [138]:
```
# cross validation
rfc_cv_scores = cross_val_score(rfc, X_train_prey, y_train_prey, cv=5,
rfc_cv_scores
```

Out[138]:
```
array([0.86401861, 0.84183217, 0.85274647, 0.83843264, 0.84574087])
```

In [139]:
```
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'])
```

Out[139]:

max_depth	2	4	6	8	10
mean_train_scores	0.77	0.83	0.87	0.92	0.96
mean_test_scores	0.77	0.83	0.87	0.92	0.96

In [140]:
```
# 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
```

```
random forest best hyperparams      : {'max_depth': 6, 'n_estimators'
: 10}
random forest best mean cv accuracy : 0.88
```

In [141]:
```
rfc_best = RandomForestClassifier(max_depth=6,n_estimators=10)
rfc_best.fit(X_train_prey, y_train_prey)
```

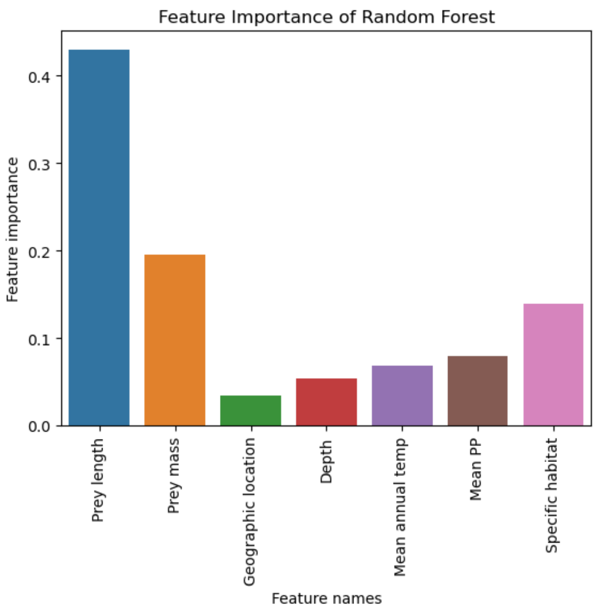
Out[141]:
```
RandomForestClassifier(max_depth=6, n_estimators=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [142]:
```
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
```

Out[142]:
```
[('Prey length', 0.42992401439283284),
 ('Prey mass', 0.19545714292515956),
 ('Specific habitat', 0.1390129260345091),
 ('Mean PP', 0.07958215448178453),
 ('Mean annual temp', 0.06859962154022503),
 ('Depth', 0.05328283389685144),
 ('Geographic location', 0.034141306728637494)]
```

In [143]:
```
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 [144]:
```
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))
```

```
The train score is: 0.802175780131692
The test score is: 0.8043509374552741
```

In [145]:
```
# 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}')
```

```
ada best hyperparams      : {'learning_rate': 1, 'n_estimators': 10}
ada best mean cv accuracy : 0.80
```

In [147]:
```
ada_best = AdaBoostClassifier(learning_rate=1,n_estimators=10)
ada_best.fit(X_train_prey, y_train_prey)
```

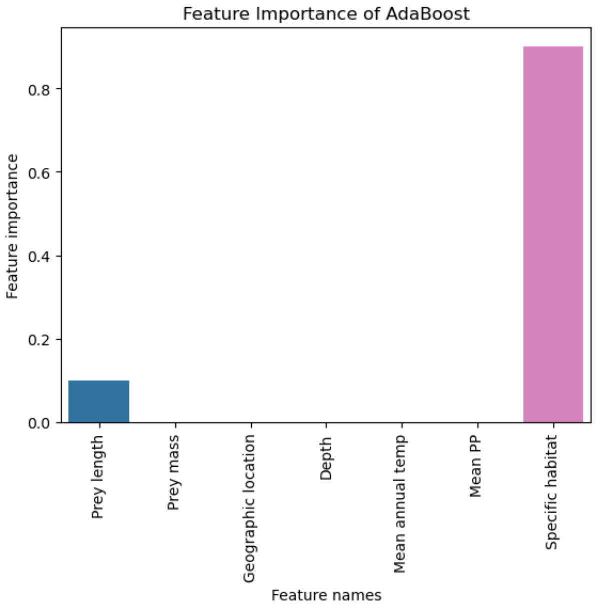
Out[147]:
```
AdaBoostClassifier(learning_rate=1, n_estimators=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [148]:
```
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
```

Out[148]:
```
[('Specific habitat', 0.9),
 ('Prey length', 0.1),
 ('Prey mass', 0.0),
 ('Geographic location', 0.0),
 ('Depth', 0.0),
 ('Mean annual temp', 0.0),
 ('Mean PP', 0.0)]
```

In [149]:
```
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

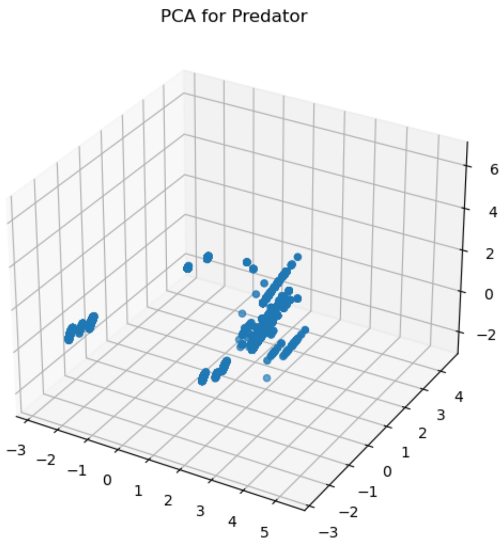
In [150]:
```
pca = PCA(n_components=3, random_state=123)

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

pca.explained_variance_ratio_
```

Out[150]:
```
array([0.3393984 , 0.25719884, 0.19599334])
```

In [151]:
```python
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')
```

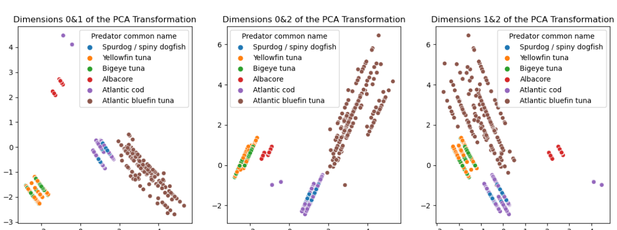
```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/1282
1549.py:4: MatplotlibDeprecationWarning: Axes3D(fig) adding itself to
the figure is deprecated since 3.4. Pass the keyword argument auto_ad
d_to_figure=False and use fig.add_axes(ax) to suppress this warning.
The default value of auto_add_to_figure will change to False in mpl3.
5 and True values will no longer work in 3.6.  This is consistent wit
h other Axes classes.
  ax = Axes3D(fig)
```

Out[151]: `Text(0.5, 0.92, 'PCA for Predator')`



In [152]:
```python
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")
```

Out[152]: `Text(0.5, 1.0, 'Dimensions 1&2 of the PCA Transformation')`



5.2 PCA for prey

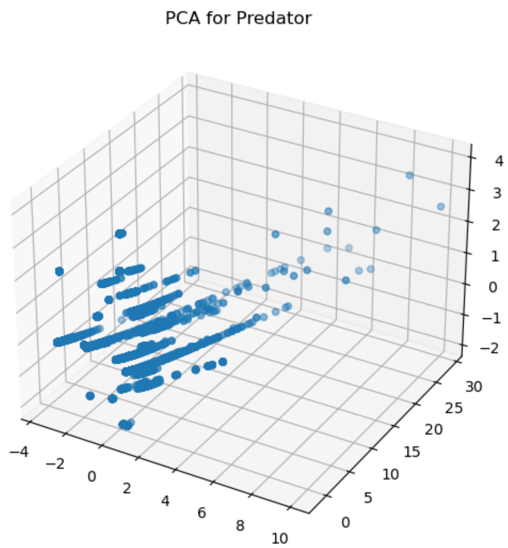
In [153]:
```python
pca = PCA(n_components=3, random_state=123)

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

pca.explained_variance_ratio_
```

Out[153]: `array([0.34895859, 0.22353736, 0.15422669])`

```
In [154]: 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')
```

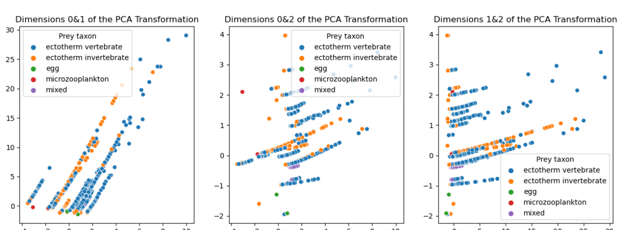
```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/4104
782933.py:2: MatplotlibDeprecationWarning: Axes3D(fig) adding itself
to the figure is deprecated since 3.4. Pass the keyword argument auto
_add_to_figure=False and use fig.add_axes(ax) to suppress this warnin
g. The default value of auto_add_to_figure will change to False in mp
l3.5 and True values will no longer work in 3.6.  This is consistent
with other Axes classes.
  ax = Axes3D(fig)
```

```
Out[154]: Text(0.5, 0.92, 'PCA for Predator')
```




```
In [155]: 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")
```

```
Out[155]: Text(0.5, 1.0, 'Dimensions 1&2 of the PCA Transformation')
```

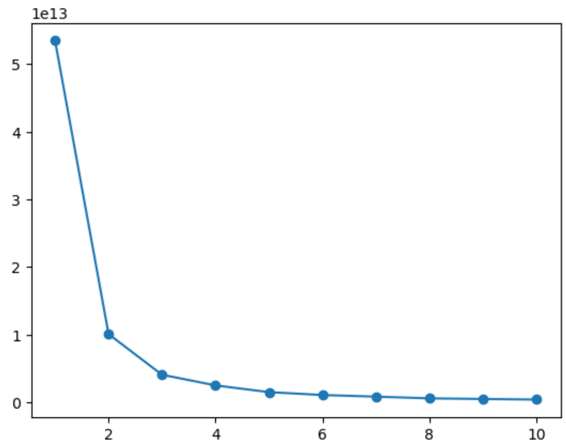


6 Clustering Model: Kmeans

6.1 Kmeans to cluster predator

In [156]:
```
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')
```

Out[156]: [<matplotlib.lines.Line2D at 0x7fd6c667bfa0>]



In [157]:
```
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()
```

Out[157]:
```
kmeans_label
0               14082
1                1224
dtype: int64
```

In [158]:
```
X_predator_cluster = pd.concat([predator_x, predator_y, labels_predato
X_predator_cluster.groupby('Predator common name').mean()
```

Out[158]:

Predator common name	Predator taxon	Predator lifestage	Type of feeding interaction	Predator length	Predator mass	Diet coverage	Geographic location	
Albacore	0.0	1.000000	0.336219	66.883552	6096.995309	0.000000	9.000000	4
Atlantic bluefin tuna	0.0	0.000000	0.123625	203.665584	156442.529544	0.000000	6.643793	
Atlantic cod	0.0	0.040111	0.754170	66.486644	2691.241660	0.181493	0.209293	
Bigeye tuna	0.0	0.000000	0.804004	107.865121	14839.720179	0.000000	3.000000	4
Spurdog / spiny dogfish	0.0	0.000000	0.421053	82.239428	2547.490880	0.031031	0.020688	
Yellowfin tuna	0.0	0.000000	1.228585	108.681022	26764.666540	0.000000	3.000000	4

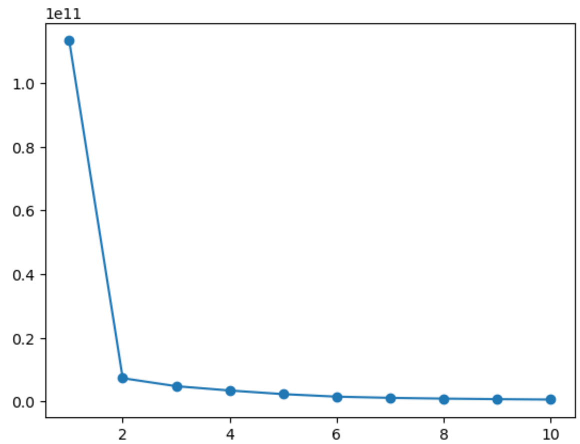
In [159]:
```
y_train_predator.value_counts()
```

Out[159]:
```
Albacore                   2895
Spurdog / spiny dogfish    2609
Atlantic cod               2003
Yellowfin tuna             1712
Atlantic bluefin tuna      1514
Bigeye tuna                1511
Name: Predator common name, dtype: int64
```

6.2 Kmeans to cluster prey

In [160]:
```
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')
```

Out[160]: [<matplotlib.lines.Line2D at 0x7fd6a27db580>]



In [161]:
```
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()
```

Out[161]:
```
kmeans_label
1               26012
0                8919
dtype: int64
```

In [162]:
```
X_prey_cluster = pd.concat([prey_x, prey_y, labels_prey], axis=1)
X_prey_cluster.groupby('Prey taxon').mean()
```

Out[162]:

	Prey length	Prey mass	Geographic location	Depth	Mean annual temp	Mean PP	Sp h
Prey taxon							
ectotherm invertebrate	3.560657	13.905011	8.013103	1599.555928	14.518936	726.588607	8.1
ectotherm vertebrate	11.430129	39.217246	7.630517	1531.068544	15.756414	739.622286	6.9
egg	0.030330	0.000002	5.080944	771.573356	1.017032	509.966273	7.8
microzooplankton	0.060302	0.000008	14.585779	2559.448081	18.395485	562.847630	16.5
mixed	11.351194	21.888489	4.559006	14.006211	23.652174	866.000000	0.0

7. Regression Model

In [163]:
```
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 [164]:
```
reg_y
```

Out[164]:
```
0        1525.6260
1        1591.7787
2        1831.7070
3          79.5090
4          57.3037
           ...
34926     510.7000
34927     508.9000
34928       6.6261
34929       6.6261
34930     369.8300
Name: Mass difference, Length: 34931, dtype: float64
```

In [165]:
```
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]]).
```

```
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/49
3454921.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
  reg_x[col_name] = reg_x[col_name].astype(str)
/var/folders/8r/f8xs7ssj5375m_ymty13gnww0000gn/T/ipykernel_46169/49
3454921.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/index
ing.html#returning-a-view-versus-a-copy)
```

In [166]:
```
# 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 [167]:
```
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))
```

```
The train score: 0.5808598730874159
The test score: 0.5563155436980785
```

7.2 Ridge Regression

In [168]:
```
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))
```

```
The train score: 0.5807786367386404
The test score: 0.5561781472999623
```

In [169]:
```
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}')
```

```
ada best hyperparams      : {'alpha': 0.1}
ada best mean cv accuracy : 0.58
```

7.3 Lasso Regression

In [170]:
```
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))
```

```
The train score: 0.5787062448058675
The test score: 0.5531628009202632
```

```
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 1.241e+13, tolerance: 6.062e+09
  model = cd_fast.enet_coordinate_descent(
```

In [171]:
```
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}')
```

```
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.647e+12, tolerance: 4.066e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.664e+12, tolerance: 4.118e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.055e+12, tolerance: 3.940e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.764e+12, tolerance: 4.118e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.075e+12, tolerance: 3.940e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.787e+12, tolerance: 4.118e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.625e+12, tolerance: 4.066e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 7.968e+12, tolerance: 3.940e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.530e+12, tolerance: 4.066e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.540e+12, tolerance: 4.118e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 7.860e+12, tolerance: 3.940e+09
  model = cd_fast.enet_coordinate_descent(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 8.412e+12, tolerance: 4.066e+09
  model = cd_fast.enet_coordinate_descent(
```

```
ada best hyperparams      : {'alpha': 0.01}
ada best mean cv accuracy : 0.58
```

```
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWar
ning: Objective did not converge. You might want to increase the numb
er of iterations, check the scale of the features or consider increas
ing regularisation. Duality gap: 1.276e+13, tolerance: 6.062e+09
  model = cd_fast.enet_coordinate_descent(
```

7.5 Decision Tree Regression

In [172]:
```python
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))
```

```
The train score: 0.7479117611237333
The test score: 0.7489658353079156
```

In [173]:
```python
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
```

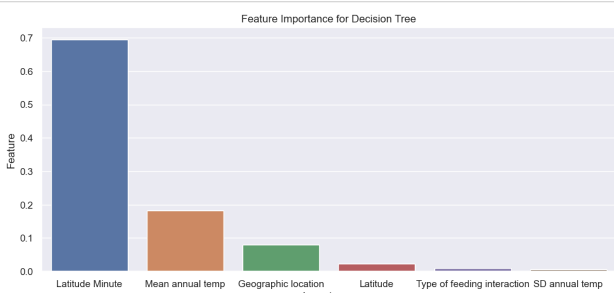
```
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-p
ackages/sklearn/tree/_classes.py:397: FutureWarning: Criterion 'mse
' was deprecated in v1.0 and will be removed in version 1.2. Use `c
riterion='squared_error'` which is equivalent.
  warnings.warn(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-p
ackages/sklearn/tree/_classes.py:397: FutureWarning: Criterion 'mse
' was deprecated in v1.0 and will be removed in version 1.2. Use `c
riterion='squared_error'` which is equivalent.
  warnings.warn(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-p
ackages/sklearn/tree/_classes.py:397: FutureWarning: Criterion 'mse
' was deprecated in v1.0 and will be removed in version 1.2. Use `c
riterion='squared_error'` which is equivalent.
  warnings.warn(
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-p
ackages/sklearn/tree/_classes.py:397: FutureWarning: Criterion 'mse
' was deprecated in v1.0 and will be removed in version 1.2. Use `c
riterion='squared_error'` which is equivalent.
```

In [174]:
```python
df = pd.DataFrame(dtr.feature_importances_, index=reg_x.columns).sort_
df['index'] = df['index'].str.split('_').str[0]
df
```

Out[174]:

	index	0
0	Latitude Minute	0.695717
1	Mean annual temp	0.182461
2	Geographic location	0.079401
3	Latitude	0.022632
4	Type of feeding interaction	0.009791
5	SD annual temp	0.006831

In [175]:
```python
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 [176]:
```python
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))
```

```
The train score: 0.5799527547426957
The test score: 0.5931023110899205
```

In [177]:
```
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}')
```

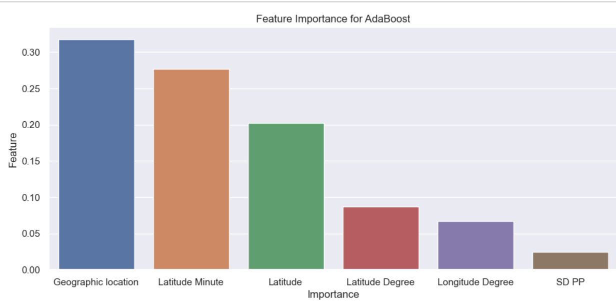
```
/Users/jingyi/opt/anaconda3/envs/MyCodingSpace/lib/python3.9/site-pac
kages/joblib/externals/loky/process_executor.py:702: UserWarning: A w
orker stopped while some jobs were given to the executor. This can be
caused by a too short worker timeout or by a memory leak.
  warnings.warn(

ada best hyperparams      : {'learning_rate': 0.01, 'n_estimators': 1
50}
ada best mean cv accuracy : 0.71
```

In [178]:
```
df = pd.DataFrame(adarg.feature_importances_, index=reg_x.columns).sor
df['index'] = df['index'].str.split('_').str[0]
df
```

Out[178]:

	index	0
0	Geographic location	0.318066
1	Latitude Minute	0.277111
2	Latitude	0.202147
3	Latitude Degree	0.086625
4	Longitude Degree	0.067136
5	SD PP	0.024572

In [179]:
```
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 [181]:
```
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'])
```

```
The train score: 0.7479117611237333
The test score: 0.7489658353079156
```

Out[181]:

max_depth	2	4	6	8	10
mean_train_scores	0.41	0.72	0.75	0.75	0.75
mean_test_scores	0.40	0.72	0.74	0.74	0.74

In [182]:
```
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
```

```
random forest best hyperparams      : {'max_depth': 6, 'n_estimators'
: 10}
random forest best mean cv accuracy : 0.74
```

In [183]:
```
df = pd.DataFrame(rf.feature_importances_, index=reg_x.columns).sort_v
df['index'] = df['index'].str.split('_').str[0]
df
```

Out[183]:

	index	0
0	Latitude Minute	0.570342
1	Mean annual temp	0.101127
2	Geographic location	0.095712
3	SD PP	0.049368
4	SD annual temp	0.041918
5	Longitude	0.037413

In [184]:
```
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()
```