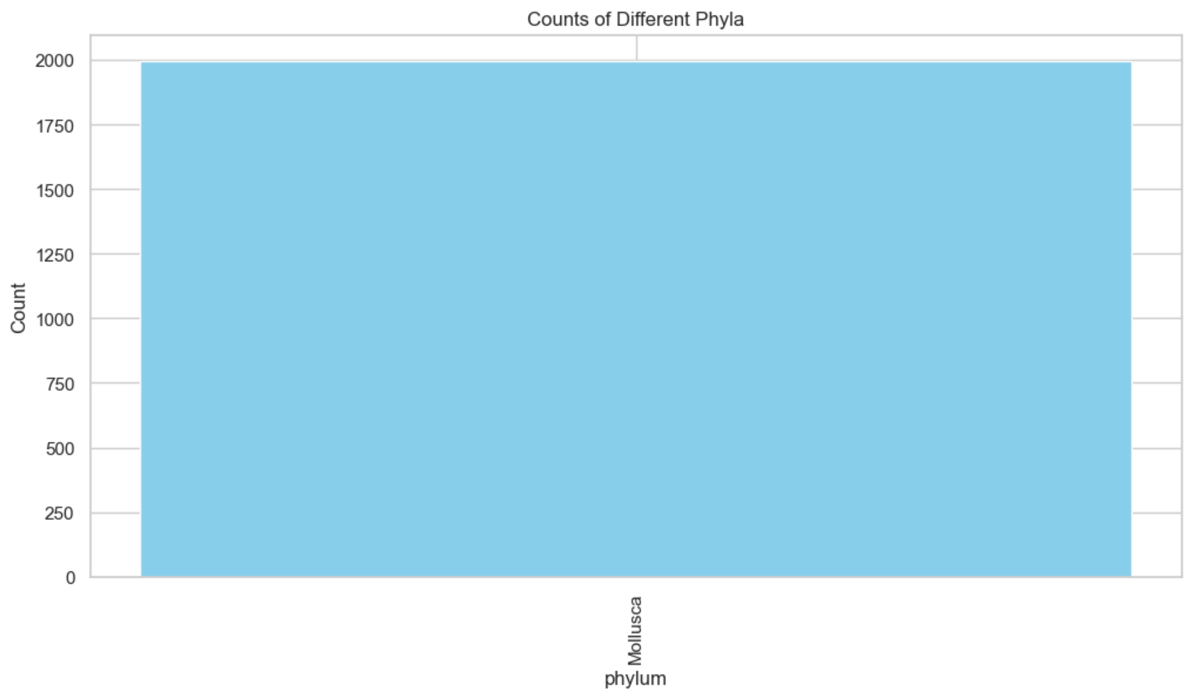
In [112…

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


df = pd.read_csv('Med_revised.csv')


species_counts = df['scientificname'].value_counts()


#for species, count in species_counts.items():
#    print(f"Species: {species}, Count: {count}")
```

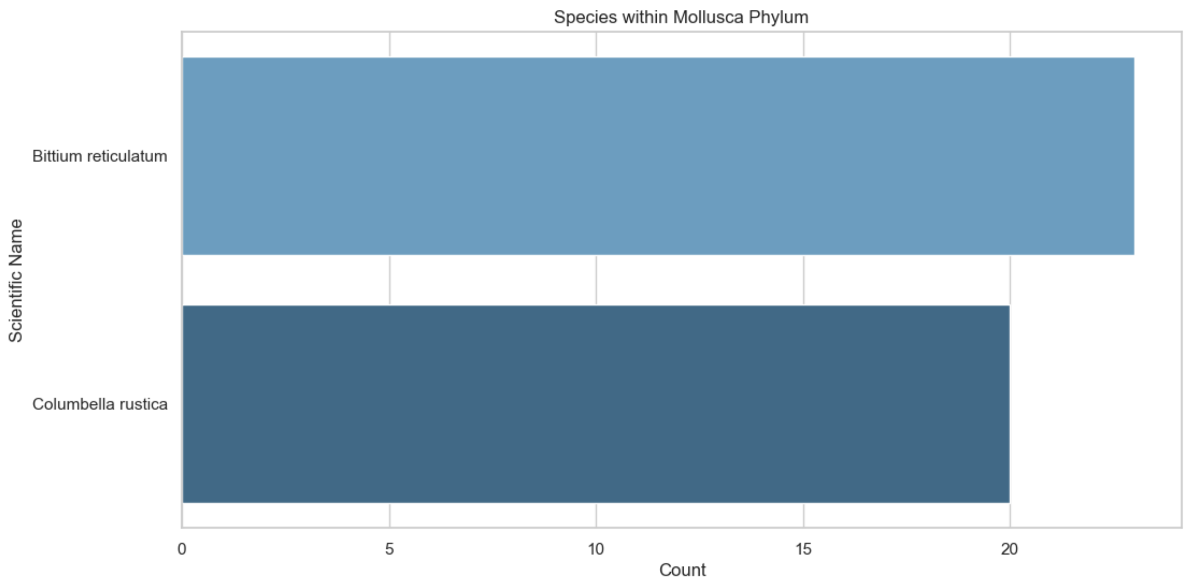
In [113…

```
# Group the data by 'Phylum' and count the occurrences of each phylum
phylum_counts = df['phylum'].value_counts().reset_index()
phylum_counts.columns = ['Phylum', 'Count']


plt.figure(figsize=(12, 6))
plt.bar(phylum_counts['Phylum'], phylum_counts['Count'], color='skyblue')
plt.xlabel('phylum')
plt.ylabel('Count')
plt.title('Counts of Different Phyla')
plt.xticks(rotation=90)
plt.show()
```



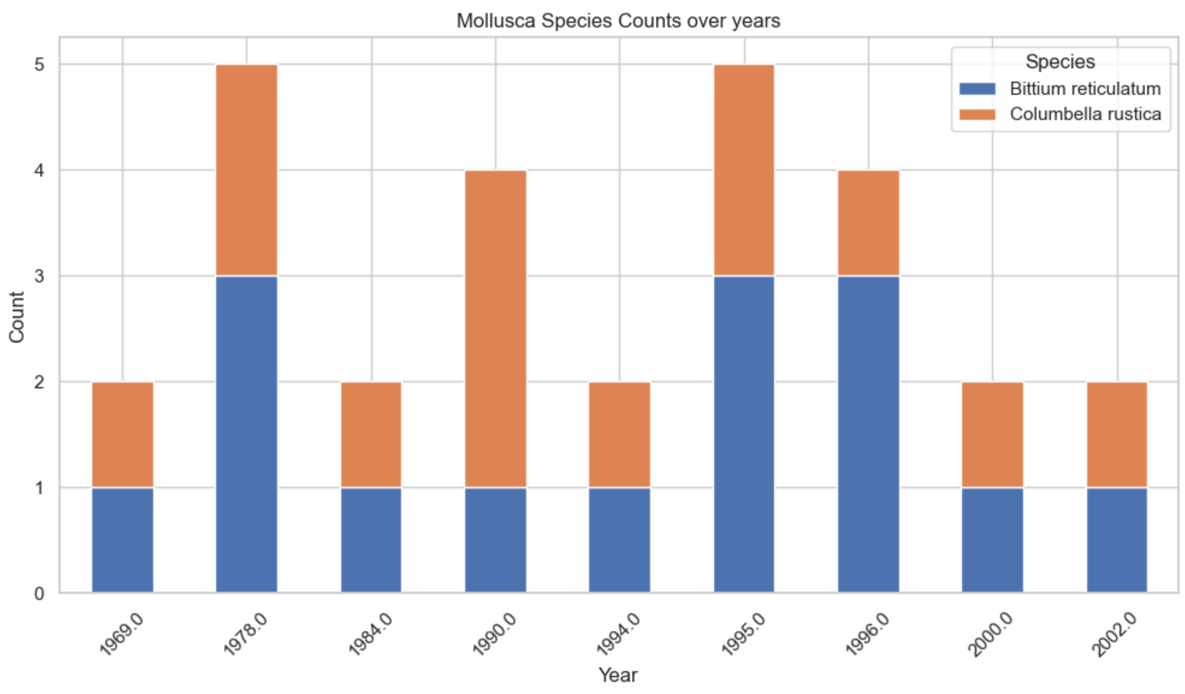
In [114…

```
# Filter the DataFrame for rows where 'Phylum' is 'Mollusca'
mollusca_data = df[df['phylum'] == 'Mollusca']

# Count the occurrences of each unique scientific name within the 'Mollusca
scientificname_counts = mollusca_data['scientificname'].value_counts().reset
scientificname_counts.columns = ['Scientific Name', 'Count']


scientificname_counts = scientificname_counts[scientificname_counts['Count']


scientificname_counts = scientificname_counts.sort_values(by='Count', ascend
plt.figure(figsize=(12, 6))
sns.set_theme(style="whitegrid")
ax = sns.barplot(x='Count', y='Scientific Name', data=scientificname_counts,
ax.set_xlabel('Count')
ax.set_ylabel('Scientific Name')
ax.set_title('Species within Mollusca Phylum')
plt.savefig("species_counts_plot_Med_Sea.pdf", format="pdf")
plt.show()
```



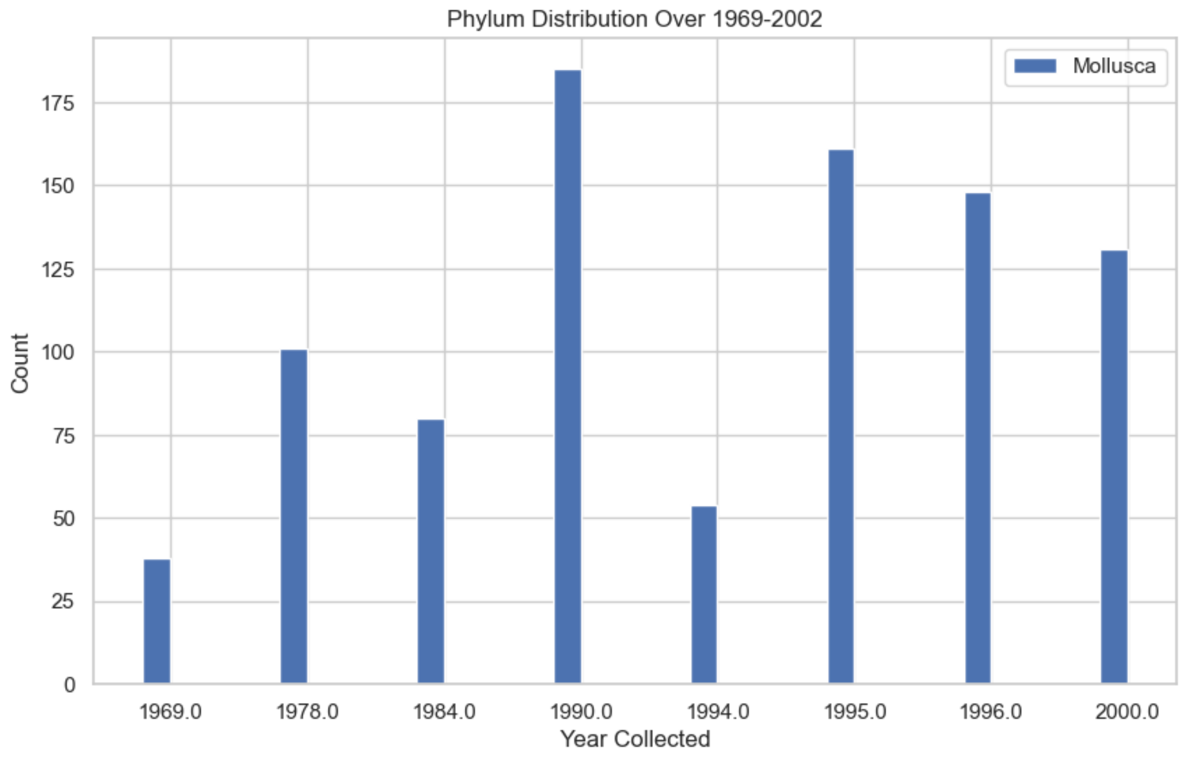
In [115…

```
years_to_analyze = [1969, 1978, 1984, 1990, 1994, 1995, 1996, 2000, 2002]


species_to_plot = ['Bittium reticulatum', 'Columbella rustica']


filtered_data = df[df['yearcollected'].isin(years_to_analyze) & df['scientif

# Group the data by species and year, and count the occurrences
species_counts = filtered_data.groupby(['yearcollected', 'scientificname']).


species_counts.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.xlabel('Year')
plt.ylabel('Count')
plt.title('Mollusca Species Counts over years')
plt.xticks(rotation=45)
plt.legend(title='Species')
plt.savefig("Mollusca_med_over_year.pdf", format="pdf")
plt.show()
```



In [126…

```
import numpy as np


filtered_df['phylum'] = filtered_df['phylum'].str.strip()

# Group the data by 'Year Collected' and 'Phylum' and calculate the count
grouped_data = filtered_df.groupby(['yearcollected', 'phylum']).size().unsta


years = grouped_data.index.astype(str)


plt.figure(figsize=(10, 6))
bar_width = 0.2
for i, phylum in enumerate(selected_phyla):
    count = grouped_data[phylum]
    x_positions = np.arange(len(years)) + i * bar_width
    plt.bar(x_positions, count, width=bar_width, label=phylum)

plt.title('Phylum Distribution Over 1969-2002')
plt.xlabel('Year Collected')
plt.ylabel('Count')
plt.xticks(np.arange(len(years)) + (bar_width * len(selected_phyla)) / 2, ye
plt.legend()
plt.show()
```



In [116…

```python
# Filter the DataFrame for rows where 'scientificname' matches either of the
selected_species = ['Bittium reticulatum', 'Columbella rustica']
abundance_data = df[df['scientificname'].isin(selected_species)]

# Calculate the total abundance for each species
abundance_counts = abundance_data['scientificname'].value_counts()


for species, count in abundance_counts.items():
    print(f"Species: {species}, Abundance: {count}")
```

```
Species: Bittium reticulatum, Abundance: 23
Species: Columbella rustica, Abundance: 20
```

In [72]:

```python
import math


abundance_bittium = 23
abundance_columbella = 20

# Calculate the total abundance in the sample
total_abundance = abundance_bittium + abundance_columbella

# Calculate the relative abundance (Pi) for each species
relative_abundance_bittium = abundance_bittium / total_abundance
relative_abundance_columbella = abundance_columbella / total_abundance

# Calculate the natural logarithm (ln) of each Pi value
ln_bittium = math.log(relative_abundance_bittium)
ln_columbella = math.log(relative_abundance_columbella)

# Calculate Pi * ln(Pi) for each species
pi_ln_bittium = relative_abundance_bittium * ln_bittium
pi_ln_columbella = relative_abundance_columbella * ln_columbella

# Calculate the Shannon Diversity Index (H) for this sample
shannon_index = - (pi_ln_bittium + pi_ln_columbella)
```

```python
print(f"Shannon Diversity Index (H): {shannon_index:.3f}")
```

```
Shannon Diversity Index (H): 0.691
```

In [ ]:
```python
import pandas as pd

df_train = pd.read_csv('Med_sea.csv')


selected_columns = df_train[['seasoncollected', 'phylum', 'yearcollected']]


output_csv_file = 'med_model_train.csv'


selected_columns.to_csv(output_csv_file, index=False)
```

In [117…
```python
season_mapping = {"spring": 1, "summer": 0, "winter":2}
df['seasoncollected'] = df['seasoncollected'].map(season_mapping)

print(season_mapping)
```

```
{'spring': 1, 'summer': 0, 'winter': 2}
```

In [118…
```python
df_m = pd.read_csv('med_model_train.csv')
df_m.head(3)

print(df_m.columns)
```

```
Index(['SC', 'seasoncollected', 'phylum_binary', 'phylum', 'yearcollecte
d'], dtype='object')
```

In [119…
```python
phylum_mapping = {'Annelida': 0, 'Anthropoda': 1, 'Mollusca': 2}

df_m['phylum_binary'] = df_m['phylum'].map(phylum_mapping)


print(df_m.head())
```

```
   SC seasoncollected  phylum_binary    phylum  yearcollected
0   1          summer              2  Mollusca           1994
1   1          summer              2  Mollusca           1994
2   1          summer              2  Mollusca           1994
3   1          summer              2  Mollusca           1994
4   1          summer              2  Mollusca           1994
```

In [120…
```python
# Drop rows with NaN values in the 'phylum_binary' column
df_m = df_m.dropna(subset=['phylum_binary'])

# Map 'seasoncollected' values to 0 for 'spring' and 1 for 'summer'
season_mapping = {'spring': 0, 'summer': 1}
df_m['SC'] = df_m['seasoncollected'].map(season_mapping)

# Convert 'phylum_binary' column to integers
df_m['phylum_binary'] = df_m['phylum_binary'].astype(int)

print(df_m.head())
```

```
    SC seasoncollected  phylum_binary    phylum  yearcollected
0  1.0          summer              2  Mollusca           1994
1  1.0          summer              2  Mollusca           1994
2  1.0          summer              2  Mollusca           1994
3  1.0          summer              2  Mollusca           1994
4  1.0          summer              2  Mollusca           1994
```

In [121…
```
selected_phyla = ['Mollusca', 'Annelida', 'Anthropoda']
df_m = df_m[df_m['phylum'].isin(selected_phyla)]
df_m.drop(columns=['seasoncollected', 'phylum'], inplace=True)
df_m.head()
```

Out[121]:

	SC	phylum_binary	yearcollected
0	1.0	2	1994
1	1.0	2	1994
2	1.0	2	1994
3	1.0	2	1994
4	1.0	2	1994

In [123…
```
# Remove rows with NaN values
df_m.dropna(subset=['phylum_binary', 'SC'], inplace=True)

# Convert the 'SC' column to integers
df_m['SC'] = df_m['SC'].astype(int)

df_m.head()
```

Out[123]:

	SC	phylum_binary	yearcollected
0	1	2	1994
1	1	2	1994
2	1	2	1994
3	1	2	1994
4	1	2	1994

In [124…
```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

# Separate the target variable from the features
X = df_m.drop(columns=['yearcollected'])
y = df_m['yearcollected']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Create and train the SVM model
clf = SVC(kernel='linear', C=1.0)
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)


from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6631578947368421
```

In [128…
```
from scipy.stats import pearsonr


# Calculate the Pearson correlation coefficient and the p-value
```

```
correlation, p_value = pearsonr(df_m['phylum_binary'], df_m['yearcollected'

print(f"Pearson Correlation: {correlation:.2f}")
print(f"P-Value: {p_value:.2f}")
```

```
Pearson Correlation: nan
P-Value: nan
```

```
/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py:4424: C
onstantInputWarning: An input array is constant; the correlation coefficien
t is not defined.
  warnings.warn(stats.ConstantInputWarning(msg))
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```