Yashvi Goyal 22BCE3019

Choice of Fishing Mode

Import the required libraries and the dataset

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

```
# URL of the CSV file
url = 'https://raw.githubusercontent.com/salemprakash/EDA/main/Data/Fishing.csv'

# Read the CSV file directly from the URL
df = pd.read_csv(url)

# Display the first few rows of the DataFrame
print(df.head())
```

```
   rownames     mode    price   catch   pbeach    ppier    pboat  pcharter  \
0         1  charter  182.930  0.5391  157.930  157.930  157.930   182.930
1         2  charter   34.534  0.4671   15.114   15.114   10.534    34.534
2         3     boat   24.334  0.2413  161.874  161.874   24.334    59.334
3         4     pier   15.134  0.0789   15.134   15.134   55.930    84.930
4         5     boat   41.514  0.1082  106.930  106.930   41.514    71.014

   cbeach   cpier   cboat  ccharter     income
0  0.0678  0.0503  0.2601    0.5391  7083.3317
1  0.1049  0.0451  0.1574    0.4671  1249.9998
2  0.5333  0.4522  0.2413    1.0266  3749.9999
3  0.0678  0.0789  0.1643    0.5391  2083.3332
4  0.0678  0.0503  0.1082    0.3240  4583.3320
```

Exploring the Dataset

```
# Dimension of the dataset
dimension = df.shape

# Summary of the dataset
summary = df.describe()

# Print results
print("Dataset Dimension: ", dimension)
print("\nDataset Summary:\n", summary)
```

```
Dataset Dimension:  (1182, 13)

Dataset Summary:
           rownames        price        catch       pbeach        ppier  \
count  1182.000000  1182.000000  1182.000000  1182.000000  1182.000000
mean    591.500000    52.081975     0.389368   103.422005   103.422005
std     341.358316    53.829970     0.560596   103.641042   103.641042
min       1.000000     1.290000     0.000200     1.290000     1.290000
25%     296.250000    15.870000     0.036100    26.656500    26.656500
50%     591.500000    37.896000     0.164300    74.628000    74.628000
75%     886.750000    67.513000     0.533300   144.144000   144.144000
max    1182.000000   666.110000     2.310100   843.186000   843.186000

             pboat     pcharter       cbeach        cpier        cboat  \
count  1182.000000  1182.000000  1182.000000  1182.000000  1182.000000
mean     55.256570    84.379244     0.241011     0.162224     0.171215
std      62.713444    63.544650     0.190752     0.160390     0.209789
min       2.290000    27.290000     0.067800     0.001400     0.000200
25%      13.122000    42.896000     0.067800     0.050300     0.023300
50%      33.534000    61.607000     0.253700     0.078900     0.089700
75%      72.402000   102.774000     0.533300     0.149800     0.241300
max     666.110000   691.110000     0.533300     0.452200     0.736900

          ccharter        income
count  1182.000000   1182.000000
mean      0.629368   4099.337054
```

```
std       0.706114   2461.964060
min       0.002100    416.666680
25%       0.021900   2083.333200
50%       0.421600   3749.999900
75%       1.026600   5416.666700
max       2.310100  12499.998000
```

Handling and Cleaning Missing Data in the Dataset

```
# Check for missing data
missing_data = df.isnull().sum()

# Print results
print("Missing Data:\n", missing_data)
```

```
Missing Data:
 rownames    0
mode        0
price       0
catch       0
pbeach      0
ppier       0
pboat       0
pcharter    0
cbeach      0
cpier       0
cboat       0
ccharter    0
income      0
dtype: int64
```

Data Cleaning

```
#Remove Duplicates

df.drop_duplicates(inplace=True)

#Handling Outliers using IQR method (only numerical columns)

numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

Q1 = df[numerical_cols].quantile(0.25)
Q3 = df[numerical_cols].quantile(0.75)
IQR = Q3 - Q1

outlier_mask = ~((df[numerical_cols] < (Q1 - 1.5 * IQR)) | (df[numerical_cols] > (Q3 + 1.5 * IQR))).any(axis=1)

# Apply the mask to the original DataFrame
df = df[outlier_mask]
print(df.describe())
```

```
          rownames       price       catch      pbeach       ppier  \
count   708.000000  708.000000  708.000000  708.000000  708.000000
mean    523.213277   42.154257    0.175666   74.445113   74.445113
std     304.083581   31.352169    0.231215   65.428828   65.428828
min       2.000000    2.290000    0.000200    2.290000    2.290000
25%     268.750000   15.870000    0.021650   22.296000   22.296000
50%     521.500000   36.330000    0.078900   57.054000   57.054000
75%     763.250000   58.480000    0.253700  106.750000  106.750000
max    1175.000000  144.246000    1.090500  305.072000  305.072000

            pboat    pcharter      cbeach       cpier       cboat    ccharter  \
count  708.000000  708.000000  708.000000  708.000000  708.000000  708.000000
mean    38.689048   66.439754    0.137756    0.069872    0.084062    0.313142
std     32.829472   32.403829    0.123395    0.046581    0.081531    0.299712
min      2.290000   27.290000    0.067800    0.001400    0.000200    0.002100
25%     12.870000   42.059000    0.067800    0.045100    0.014300    0.020900
50%     27.870000   56.870000    0.067800    0.078900    0.053100    0.242100
75%     55.930000   84.760000    0.253700    0.078900    0.157400    0.539100
max    159.770000  178.270000    0.533300    0.149800    0.260100    1.090500

             income
count    708.000000
mean    3808.851023
std     1952.320717
min      416.666680
```

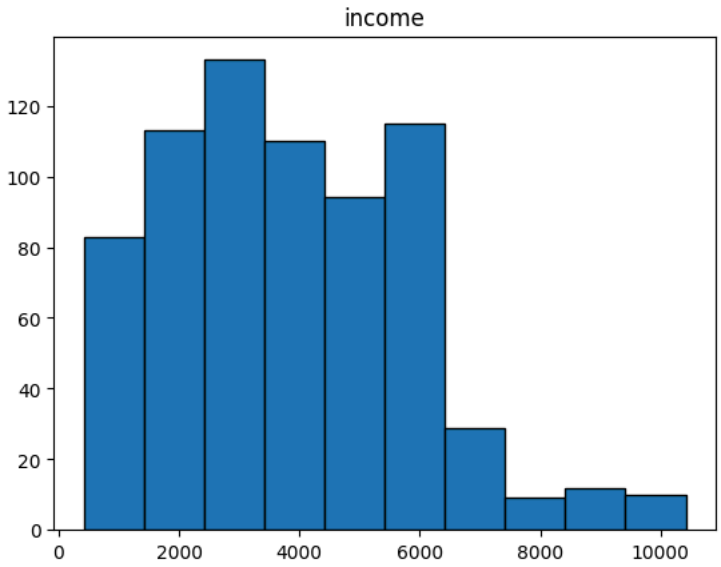
```
25%     2083.333200
50%     3749.999900
75%     4583.332000
max    10416.666000
```

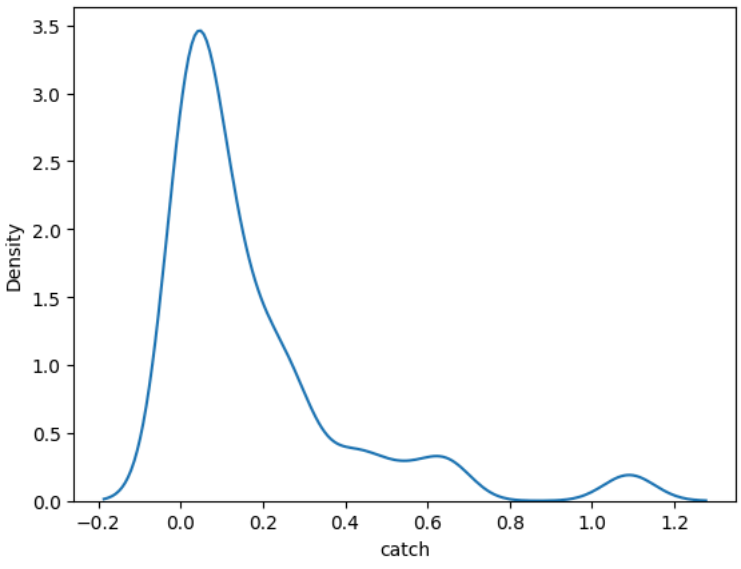
Univariate Analysis

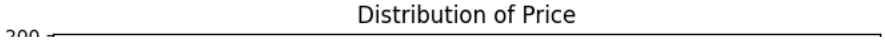
```
#Print the number of inputs for each type of mode
print(df['mode'].value_counts())

#Print the income generated by all the modes combined
df.hist(column='income', grid=False, edgecolor='black')
plt.show()
sns.kdeplot(df['catch'])
plt.show()
plt.figure(figsize=(8, 6))
sns.histplot(df['price'], kde=True, bins=10)
plt.title('Distribution of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

```
mode
boat       255
charter    250
pier       120
beach       83
Name: count, dtype: int64
```





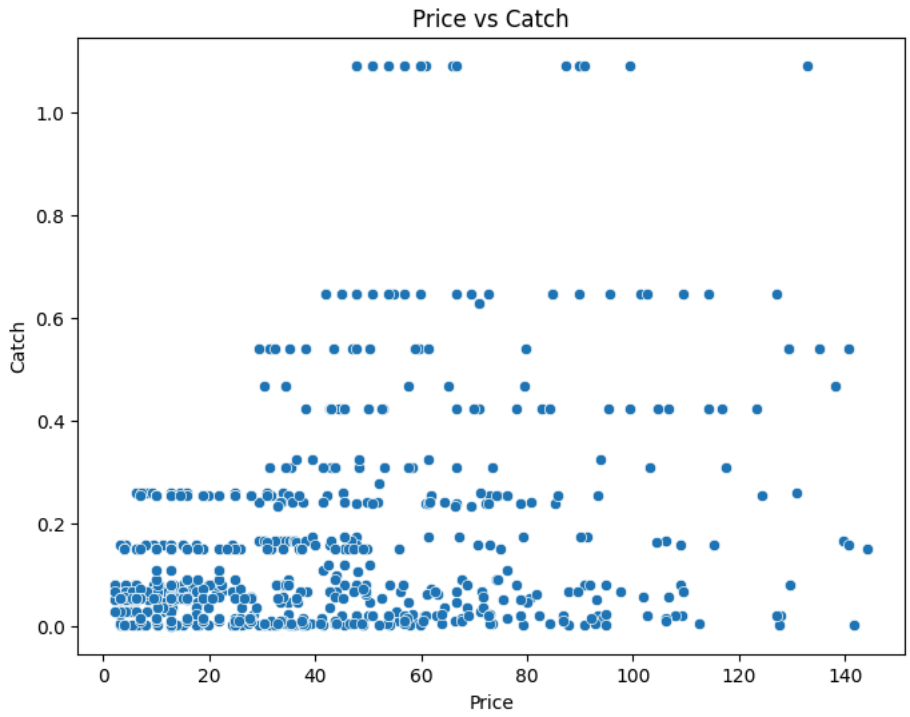


Bivariate Analysis

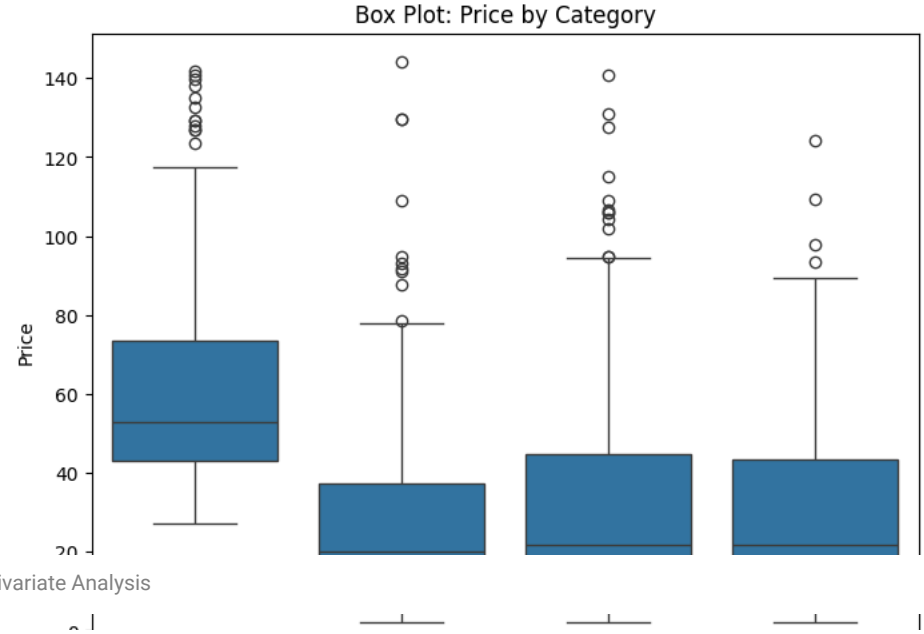
```
# Scatter plot of Price vs Catch
plt.figure(figsize=(8, 6))
sns.scatterplot(x='price', y='catch', data=df)
plt.title('Price vs Catch')
plt.xlabel('Price')
plt.ylabel('Catch')
plt.show()

# Box plot of Price by Category
plt.figure(figsize=(8, 6))
sns.boxplot(x='mode', y='price', data=df)
plt.title('Box Plot: Price by Category')
plt.xlabel('Category')
plt.ylabel('Price')
plt.show()

plt.figure(figsize=(8, 6))
corr = df[['price', 'catch']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```




```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be r
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



Multivariate Analysis

```
# Pairplot for multivariate relationships
sns.pairplot(df[['price', 'income', 'pbeach', 'cbeach']])
plt.show()

# Select the relevant numerical columns for the heatmap
numerical_cols = ['price', 'catch', 'pbeach', 'ppier', 'pboat', 'pcharter', 'cbeach', 'cpier', 'cboat', 'ccharter', 'income'

# Generate the correlation matrix for the selected columns
corr_matrix = df[numerical_cols].corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
# Melt the DataFrame to format it for the boxplot
df_melted = df.melt(value_vars=['price', 'catch', 'income', 'pbeach', 'pboat'])

# Create a box plot for the selected numerical variables
plt.figure(figsize=(10, 6))
sns.boxplot(x='variable', y='value', data=df_melted)
plt.title('Box Plot for Multiple Variables')
plt.show()


# Create a violin plot for the selected numerical variables
plt.figure(figsize=(10, 6))
sns.violinplot(x='variable', y='value', data=df_melted)
plt.title('Violin Plot for Multiple Variables')
```