```
import numpy as np
import pandas as pd

abalone = pd.read_csv("abalone.csv")

abalone.head()
```

```
  Sex  Length  Diameter  Height  Whole weight  Shucked weight  Viscera
weight  \
0   M   0.455     0.365   0.095        0.5140          0.2245
0.1010
1   M   0.350     0.265   0.090        0.2255          0.0995
0.0485
2   F   0.530     0.420   0.135        0.6770          0.2565
0.1415
3   M   0.440     0.365   0.125        0.5160          0.2155
0.1140
4   I   0.330     0.255   0.080        0.2050          0.0895
0.0395

   Shell weight  Rings
0         0.150     15
1         0.070      7
2         0.210      9
3         0.155     10
4         0.055      7
```

```
abalone.shape
```

```
(4177, 9)
```

```
abalone.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
abalone.isnull().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

```
abalone.duplicated().sum()
```

```
np.int64(0)
```

```
abalone.describe()
```

```
             Length     Diameter       Height  Whole weight  Shucked
weight  \
count  4177.000000  4177.000000  4177.000000   4177.000000
4177.000000
mean      0.523992     0.407881     0.139516      0.828742
0.359367
std       0.120093     0.099240     0.041827      0.490389
0.221963
min       0.075000     0.055000     0.000000      0.002000
0.001000
25%       0.450000     0.350000     0.115000      0.441500
0.186000
50%       0.545000     0.425000     0.140000      0.799500
0.336000
75%       0.615000     0.480000     0.165000      1.153000
0.502000
max       0.815000     0.650000     1.130000      2.825500
1.488000

       Viscera weight  Shell weight        Rings
count     4177.000000   4177.000000  4177.000000
mean         0.180594      0.238831     9.933684
std          0.109614      0.139203     3.224169
min          0.000500      0.001500     1.000000
25%          0.093500      0.130000     8.000000
50%          0.171000      0.234000     9.000000
75%          0.253000      0.329000    11.000000
max          0.760000      1.005000    29.000000
```

Encoding

```
abalone['Sex'].value_counts()
```

```
Sex
M    1528
I    1342
F    1307
Name: count, dtype: int64
```

```
abalone['Sex'] = abalone['Sex'].map({"M":0,"F":1,"I":2})
```

```
abalone['Sex'].value_counts()
```

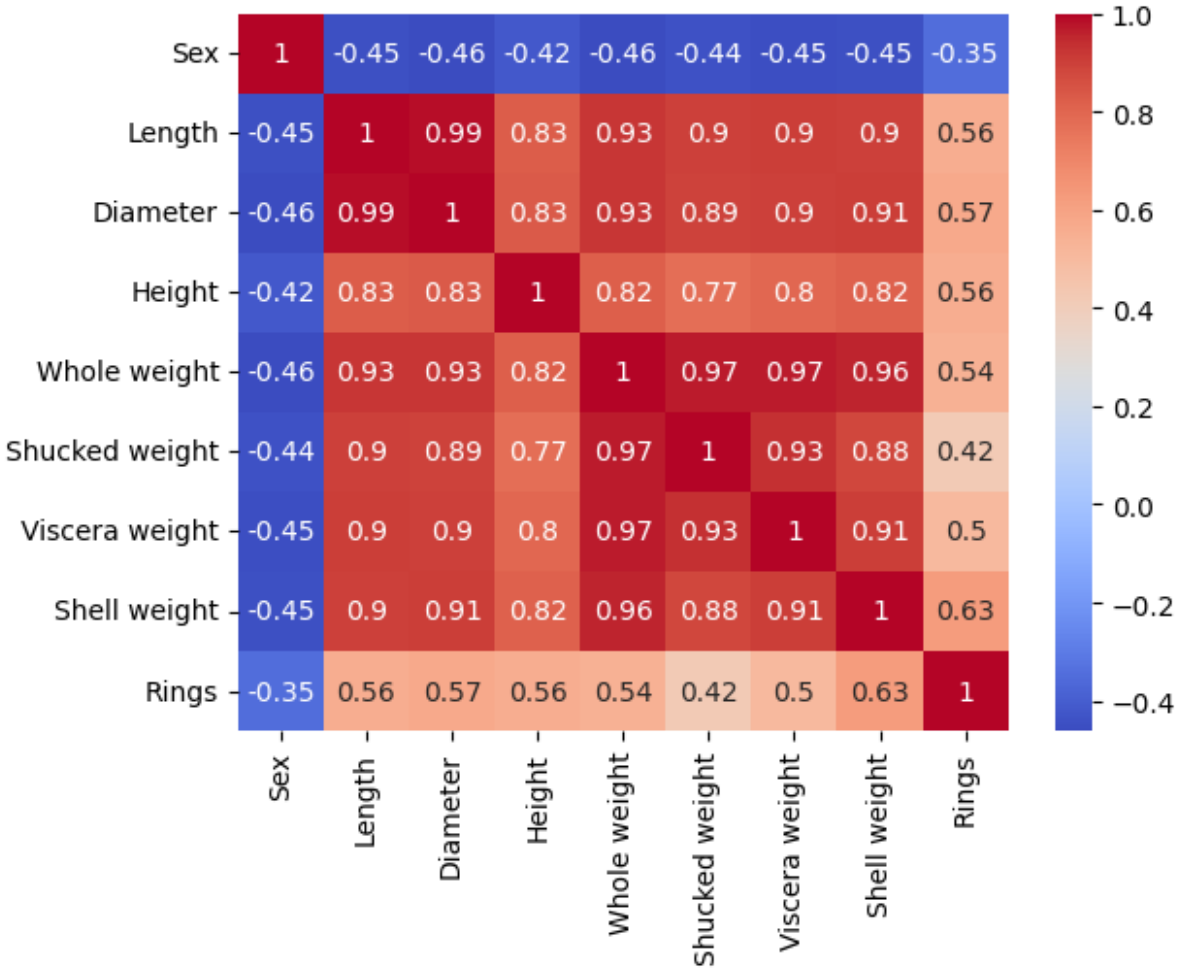
```
Sex
0    1528
2    1342
1    1307
Name: count, dtype: int64
```

EDA (Exploratory data Analysis)

```
corr = abalone.corr()
```

```
import seaborn as sns
sns.heatmap(corr,annot=True,cbar=True,cmap='coolwarm')
```

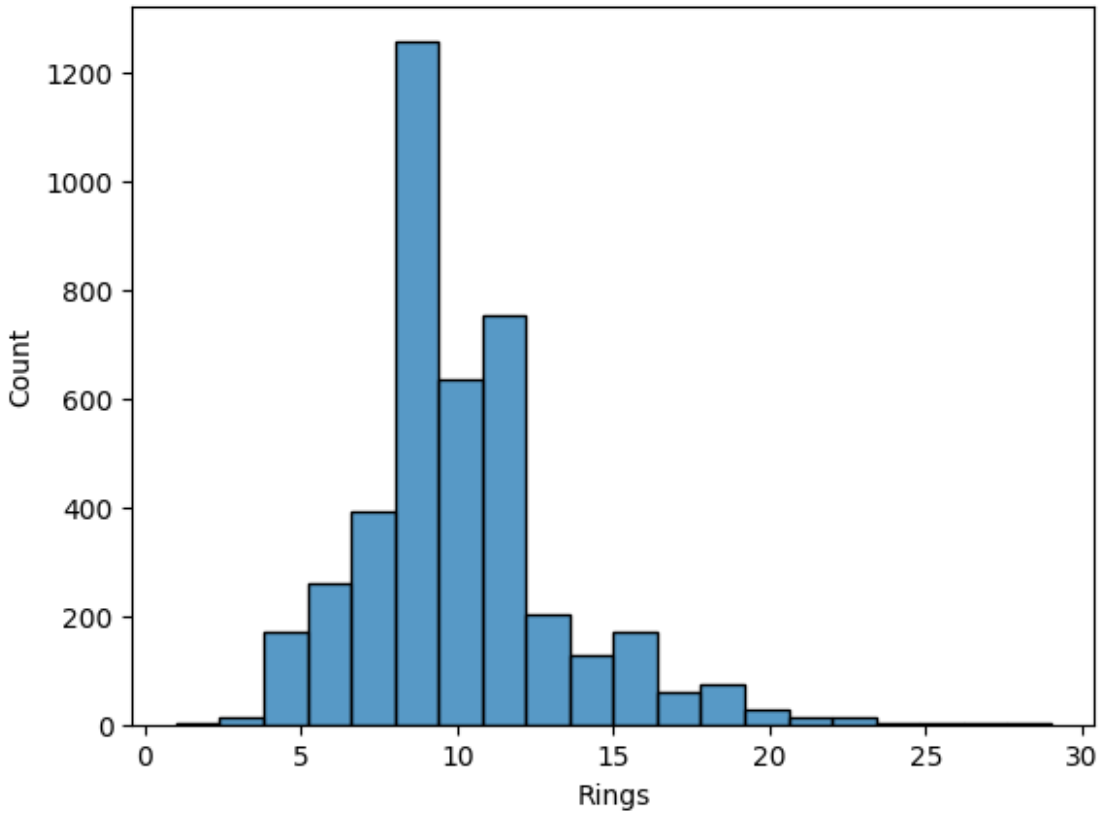
```
<Axes: >
```



Distribution of target variable (age)

```
sns.histplot(abalone['Rings'],bins=20)
```

```
<Axes: xlabel='Rings', ylabel='Count'>
```




```
abalone['Rings'].value_counts()
```

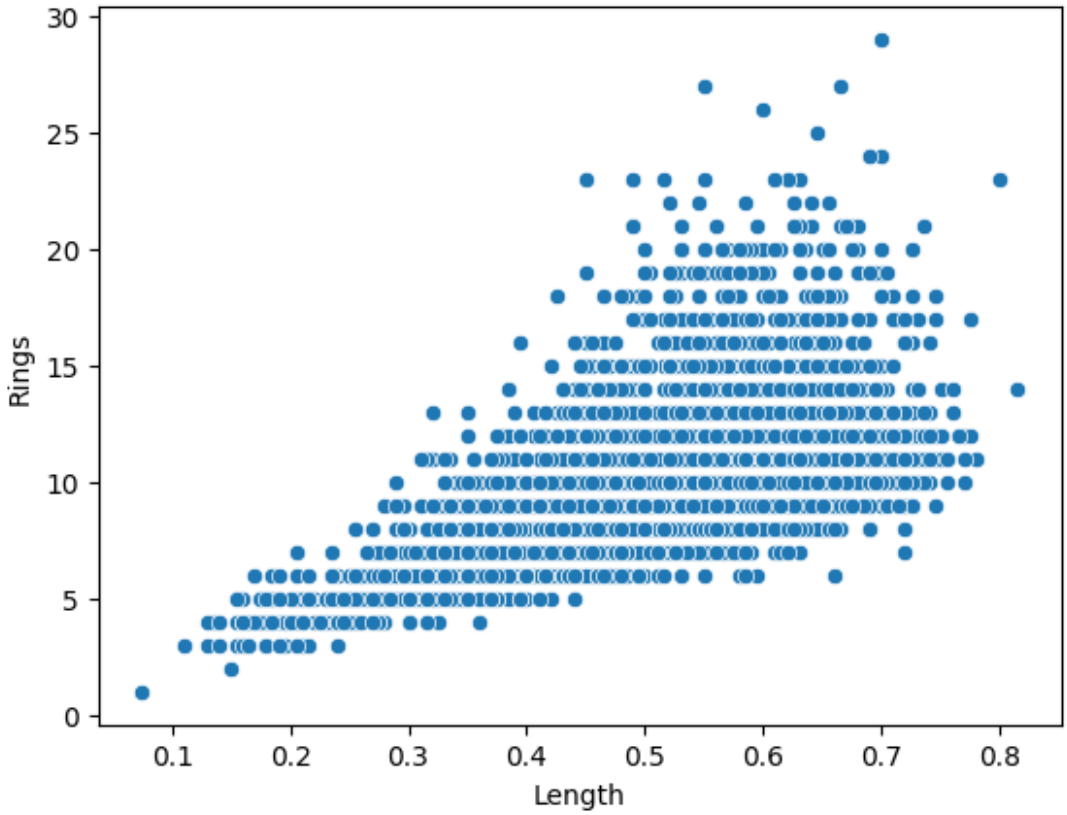
```
Rings
9     689
10    634
8     568
11    487
7     391
12    267
6     259
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
```

```
24      2
27      2
1       1
29      1
26      1
2       1
25      1
Name: count, dtype: int64
```

Scatter plot of length vs age

```
sns.scatterplot(x='Length',y='Rings',data=abalone)
```

```
<Axes: xlabel='Length', ylabel='Rings'>
```



Train Test Split

```
X = abalone.drop('Rings',axis=1)
y = abalone['Rings']

from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Standarization the data

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.transform(X_test)

X_test_scaled
```

```
array([[-1.15518338,  0.67189513,  0.47107108, ...,  0.27104784,
         1.10272193,  0.60976664],
       [-1.15518338,  0.54597132,  0.31905249, ...,  0.11857068,
         0.31224199,  0.03801081],
       [ 0.05278772,  0.29412372,  0.36972535, ..., -0.24916836,
         0.39905771,  0.68123611],
       ...,
       [ 1.26075882,  0.16819992,  0.21770676, ..., -0.03614879,
        -0.20865231, -0.22642626],
       [ 1.26075882, -0.50339368, -0.5423862 , ..., -0.47339947,
        -0.81636232, -0.39795301],
       [ 1.26075882, -1.34288568, -1.35315201, ..., -1.17748518,
        -1.30984112, -1.17697032]], shape=(836, 8))
```

```
X_test_scaled
```

```
array([[-1.15518338,  0.67189513,  0.47107108, ...,  0.27104784,
         1.10272193,  0.60976664],
       [-1.15518338,  0.54597132,  0.31905249, ...,  0.11857068,
         0.31224199,  0.03801081],
       [ 0.05278772,  0.29412372,  0.36972535, ..., -0.24916836,
         0.39905771,  0.68123611],
       ...,
       [ 1.26075882,  0.16819992,  0.21770676, ..., -0.03614879,
        -0.20865231, -0.22642626],
       [ 1.26075882, -0.50339368, -0.5423862 , ..., -0.47339947,
        -0.81636232, -0.39795301],
       [ 1.26075882, -1.34288568, -1.35315201, ..., -1.17748518,
        -1.30984112, -1.17697032]], shape=(836, 8))
```

Training Models

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Define a list of models to train and compare
models = [
    ('Linear Regression', LinearRegression()),
    ('Ridge Regression', Ridge()),
    ('Lasso Regression', Lasso()),
    ('Decision Tree', DecisionTreeRegressor(random_state=42)),
    ('Random Forest', RandomForestRegressor(random_state=42))
]

# Train and evaluate each model
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f'{name}: MSE = {mse:.2f}, R2 = {r2:.2f}')
```

```
Linear Regression: MSE = 4.95, R2 = 0.54
Ridge Regression: MSE = 4.99, R2 = 0.54
Lasso Regression: MSE = 10.83, R2 = -0.00
Decision Tree: MSE = 9.02, R2 = 0.17
Random Forest: MSE = 5.07, R2 = 0.53
```

```
# The MSE represents the average squared difference between the
predicted and actual values, and a lower MSE indicates better
performance.
# The R2 score represents the proportion of variance in the target
variable that is predictable from the independent variables, and a
higher R2 score indicates better performance.
```

Chosen Model

```
dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train)
y_pred = dtr.predict(X_test)
print(mean_squared_error(y_test, y_pred))
print(r2_score(y_test,y_pred))
```

```
9.471291866028707
0.12507084609267938
```

Prediction System

```
def prediction_age(Sex, Length, Diameter, Height, Whole_weight,
Shucked_weight, Viscera_weight, Shell_weight):
    features = pd.DataFrame([{
        'Sex': Sex,
        'Length': Length,
        'Diameter': Diameter,
        'Height': Height,
        'Whole weight': Whole_weight,
        'Shucked weight': Shucked_weight,
        'Viscera weight': Viscera_weight,
        'Shell weight': Shell_weight
    }])

    pred = dtr.predict(features)
    return pred[0]

# Sample test inputs (just examples)
Sex = 2
Length = 8.0
Diameter = 4.0
Height = 6.0
Whole_weight = 10.0
Shucked_weight = 20.0
Viscera_weight = 20.0
Shell_weight = 15.0

# Make prediction
prediction = prediction_age(Sex, Length, Diameter, Height,
Whole_weight, Shucked_weight, Viscera_weight, Shell_weight)

# Print result
print("Predicted Age (approx. number of rings):", round(prediction,
2))
```

```
Predicted Age (approx. number of rings): 14.0
```

```
import pickle
pickle.dump(dtr,open('model.pkl','wb'))

import pickle
import pandas as pd

model = pickle.load(open("model.pkl", "rb"))

sample = pd.DataFrame([{
    'Sex': 0,
    'Length': 0.455,
    'Diameter': 0.365,
```

```
    'Height': 0.095,
    'Whole weight': 0.514,
    'Shucked weight': 0.2245,
    'Viscera weight': 0.101,
    'Shell weight': 0.15
}])

pred = model.predict(sample)
print("Predicted Age:", pred)
```

```
Predicted Age: [15.]
```