Assignment 1

Problem 1

Import necessary libraries

First, we need to import basic libraries such as Pandas, NumPy, Seaborn, Matplotlib, Plotly, and cufflinks. these libraries are used accross different sections of the process

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

print(__version__) # requires version >= 1.9.0
import cufflinks as cf
# For Notebooks
init_notebook_mode(connected=True)

# For offline use
cf.go_offline()
```

```
5.18.0
```

In [2]:
```
import warnings

warnings.filterwarnings("ignore")
```

load Dataset 1

in this section, dataset will be loaded using pandas read_csv method. next, the columns will be renamed, so that we can tell apart 'Vertical', 'Diagonal' and 'Cross' lengths.

In [3]:
```
df = pd.read_csv('Dataset_I.csv')
df.columns = ['Species', 'Weight', 'Vertical', 'Diagonal', 'Cross', 'Height', 'Width']
df
```

Out[3]:

	Species	Weight	Vertical	Diagonal	Cross	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340
...	...	...	...	...	...	...	...
154	Smelt	12.2	11.5	12.2	13.4	2.0904	1.3936
155	Smelt	13.4	11.7	12.4	13.5	2.4300	1.2690
156	Smelt	12.2	12.1	13.0	13.8	2.2770	1.2558
157	Smelt	19.7	13.2	14.3	15.2	2.8728	2.0672
158	Smelt	19.9	13.8	15.0	16.2	2.9322	1.8792

159 rows × 7 columns

Part a:

Determine the number of fish species present in the dataset and analyze their distribution across each class. (Plot a bar chart too!).

to show the number of fish species, GroupBy method is used. since our dataset does not include any NaN values, the count of elements for each feature is the same. so here we only selected the count of one feature such as Weight to preview the number of each species

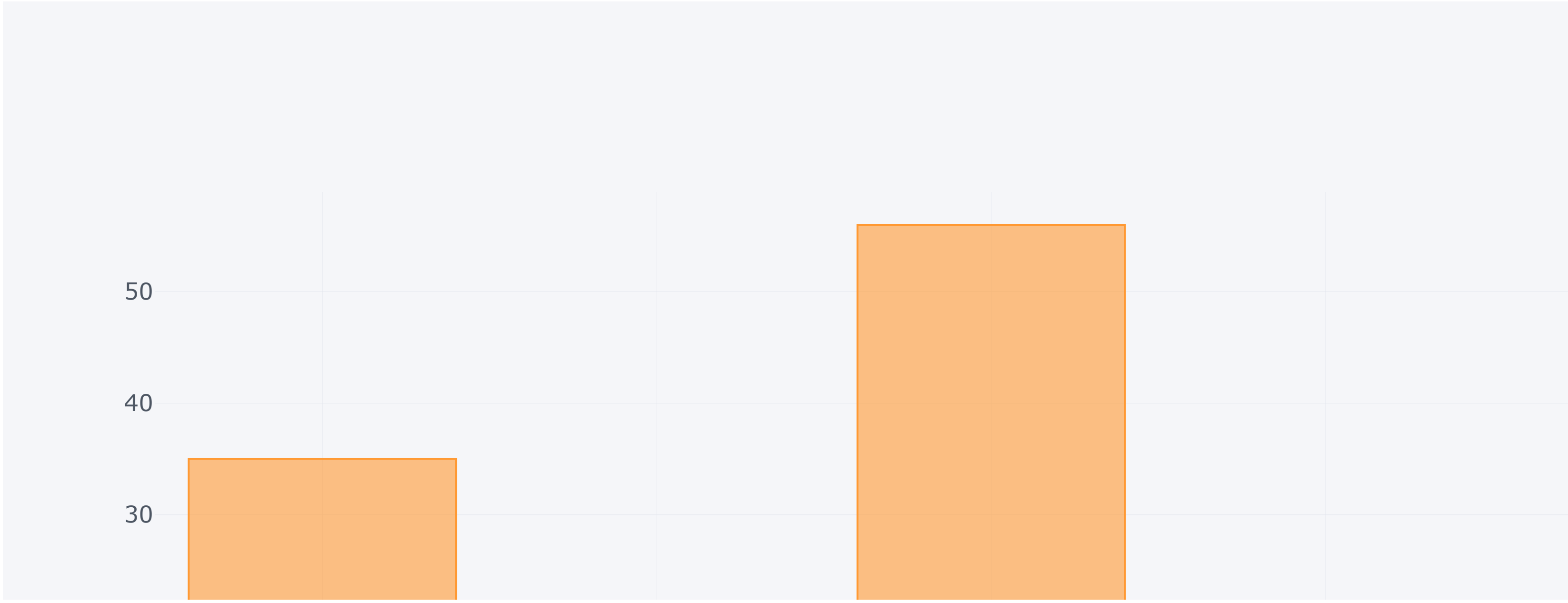
In [4]:
```
pd.DataFrame(df.groupby('Species')['Weight'].count())
```

Out[4]:

Species	Weight
Bream	35
Parkki	11
Perch	56
Pike	17
Roach	20
Smelt	14
Whitefish	6

Next, a bar chart is plotted using plotly We can see that Perch and Bream have the highest count among our dataset and WhiteFish and Parkiki have the lowest

In [5]:
```
df.groupby('Species')['Weight'].count().iplot(kind = 'bar')
```

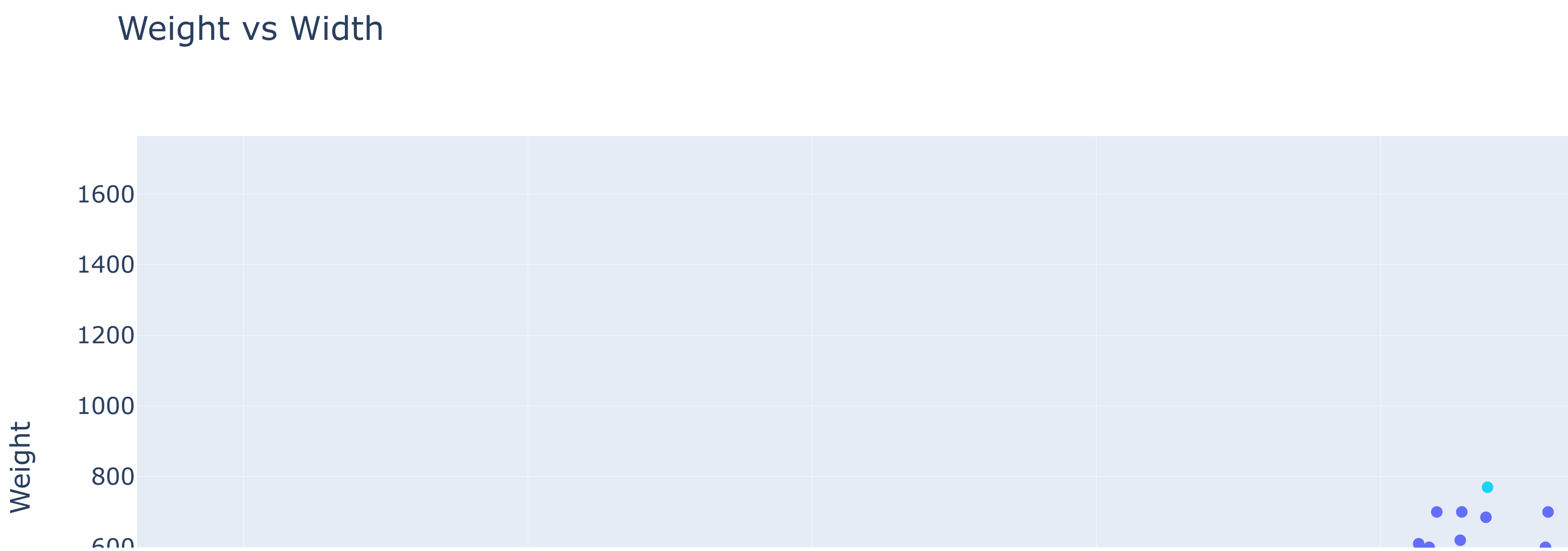


Part b:

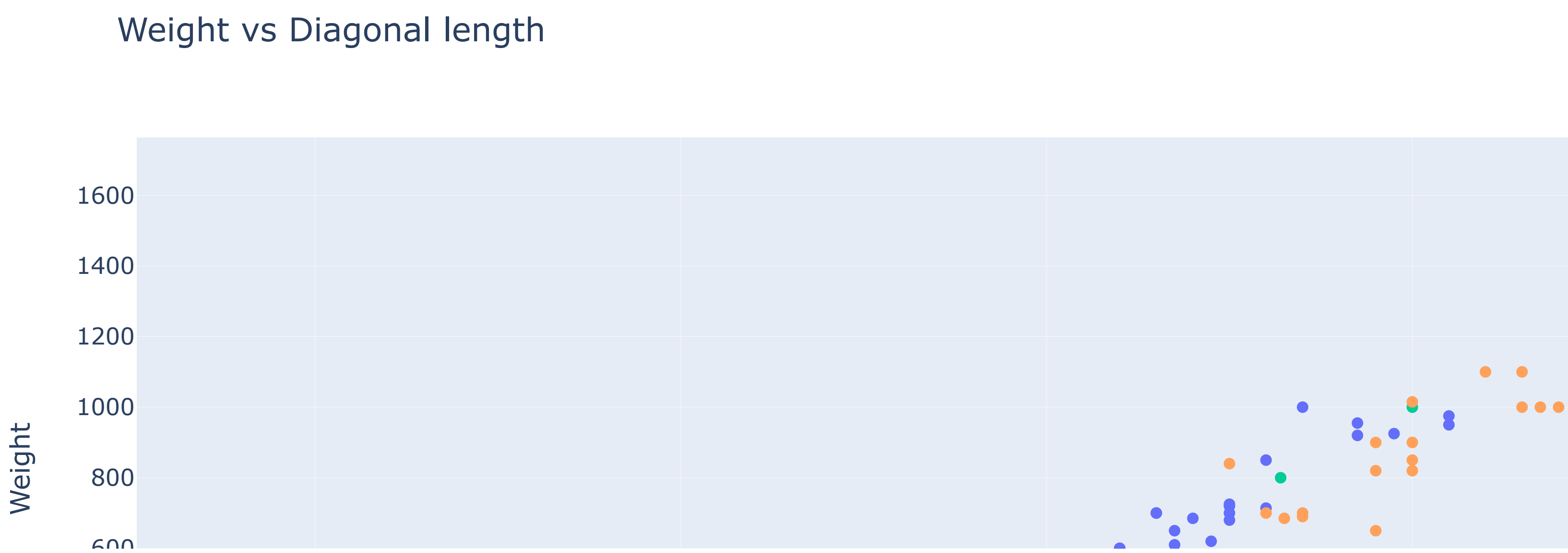
Find the relationship between the following features: (weight vs. width) ,(weight vs. diagonal length), (cross length vs. vertical length). Use scatter diagrams to visualize the data and provide a detailed explanation of your findings.

In [6]:
```
import plotly.express as px
```

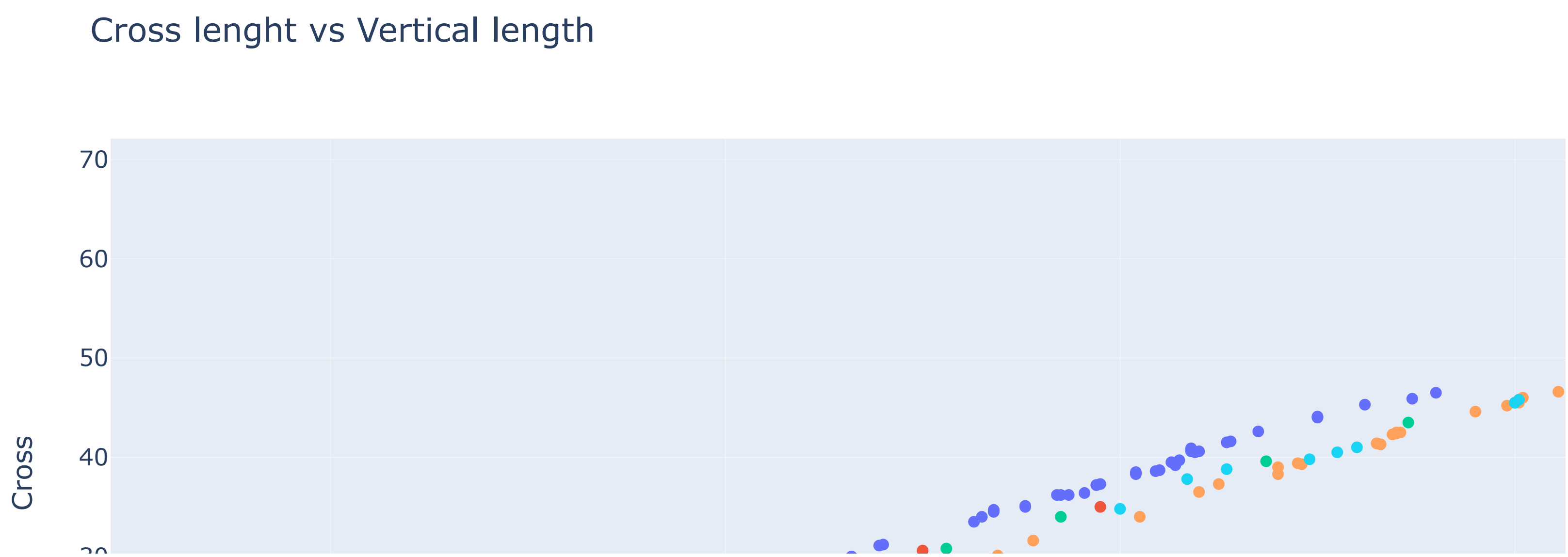
```
fig = px.scatter(df, y='Weight', x='Width', color='Species' , title='Weight vs Width')
fig.show()
```



In [7]:
```
fig = px.scatter(df, y='Weight', x='Diagonal', color='Species' , title='Weight vs Diagonal length')
fig.show()
```



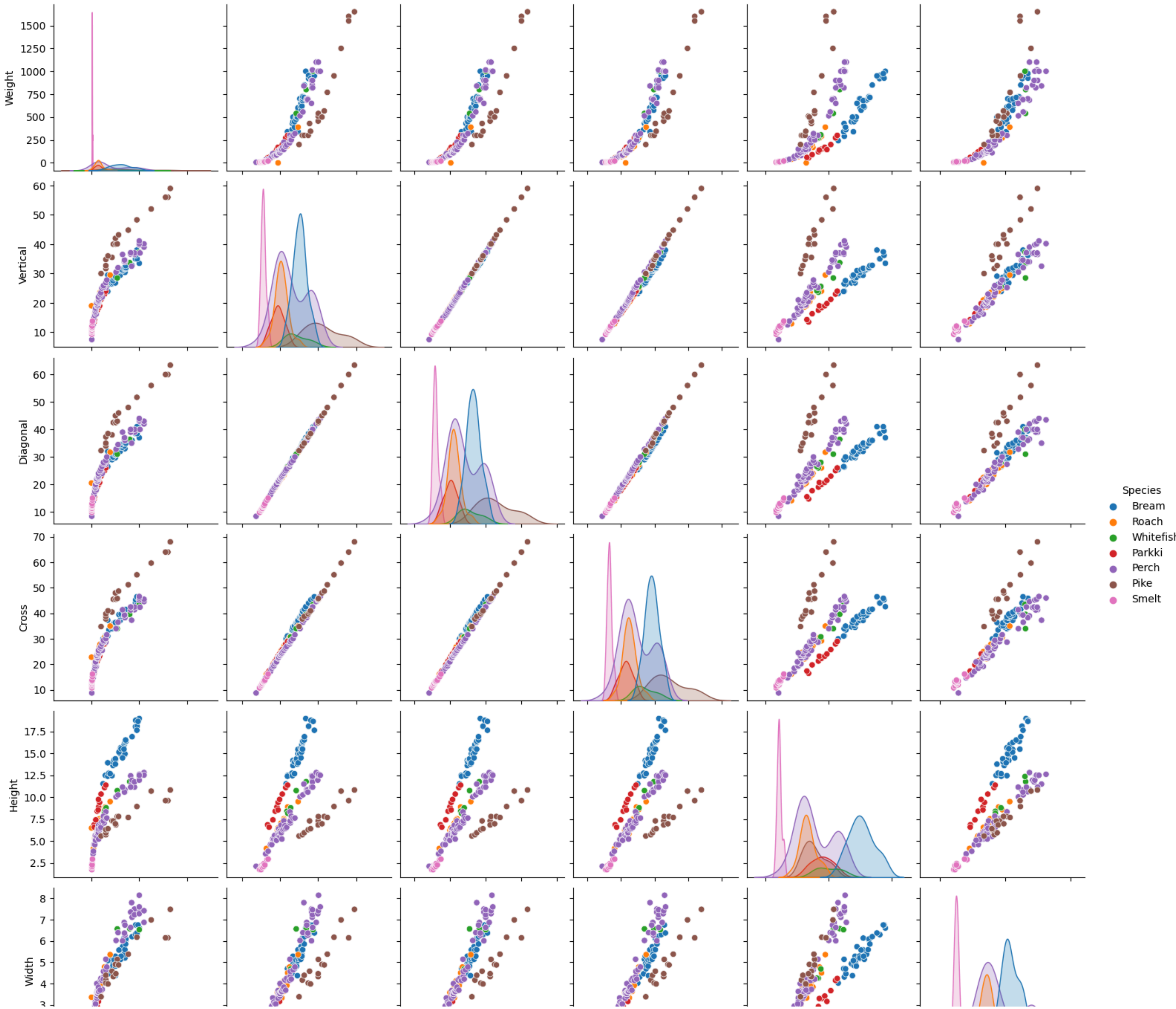
In [8]:
```
fig = px.scatter(df, y='Cross', x='Vertical', color='Species', title='Cross lenght vs Vertical length')
fig.show()
```

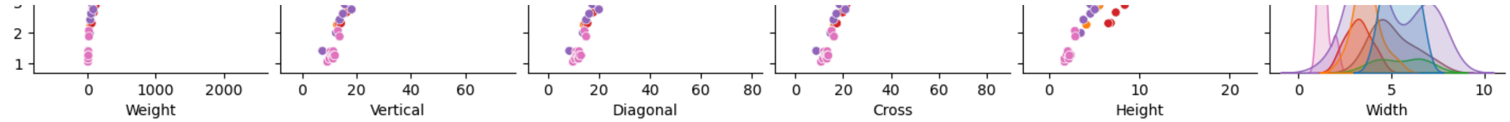


we plot the pair plot of our dataset to find any possible relationship between the features and columns

In [9]:
```
sns.pairplot(df , hue = 'Species')
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x2a6d53c4ac0>


Weight
Vertical
Diagonal
Cross
Height
Width
Species
Bream
Roach
Whitefish
Parkki
Perch
Pike
Smelt



Part c:

Develop a simple linear regression model for (cross length vs. vertical length). Fit the model and assess its performance using the mean squared error (MSE) and mean absolute error (MAE) metrics. Compare and plot the results.

In [10]:
```
X1 = df['Vertical']
Y1 = df['Cross']
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X1, Y1, test_size=0.3 , random_state=101)
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(np.array(X_train).reshape(-1,1),y_train)

# The coefficients
print('Coefficients: \n', model.coef_)
print('intercept: \n', model.intercept_)
```

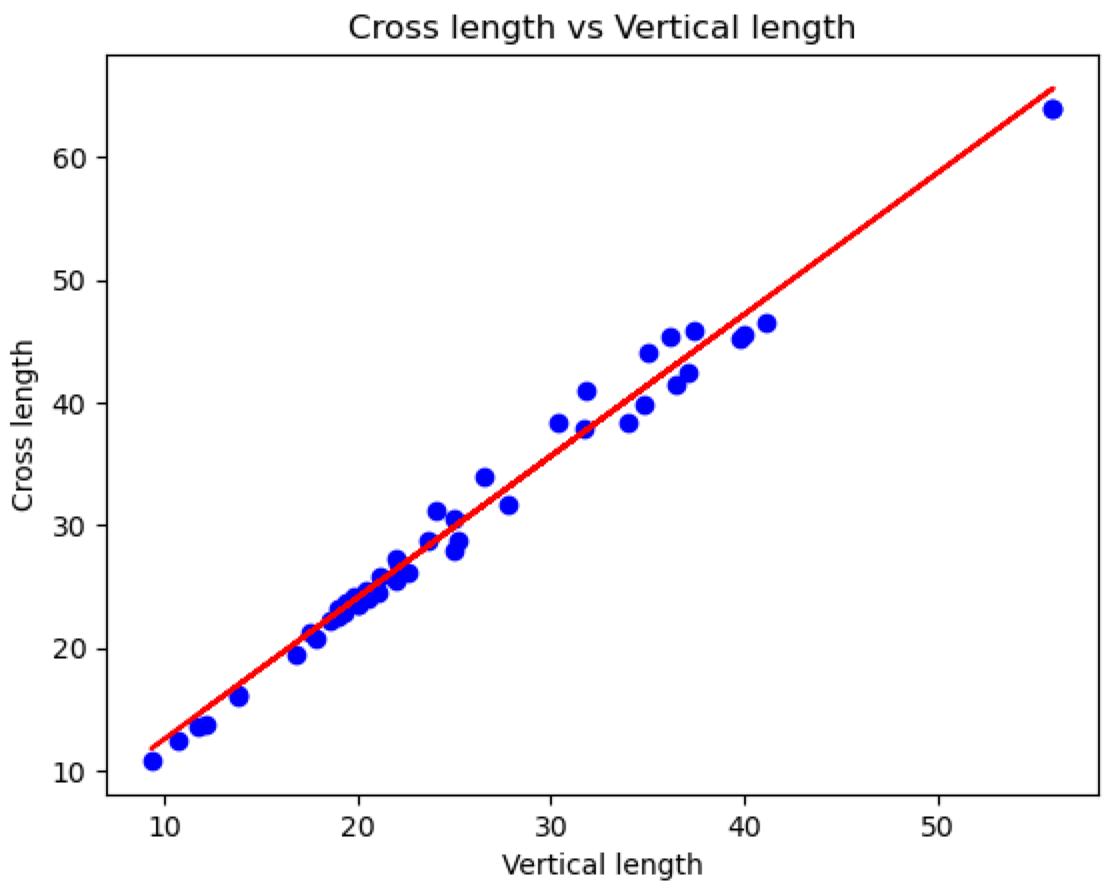
```
Coefficients: 
 [1.15362994]
intercept: 
 1.0212673228352038
```

In [11]:
```
from sklearn import metrics

predictions = model.predict( np.array(X_test).reshape(-1,1))
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 1.1205663200084255
MSE: 1.8460614768618697
RMSE: 1.358698449569245
```

In [12]:
```
plt.scatter(X_test, y_test,  color='blue')
plt.plot(X_test, model.coef_*X_test + model.intercept_, '-r')
plt.xlabel("Vertical length")
plt.ylabel("Cross length")
plt.title('Cross length vs Vertical length')
```

Out[12]:
```
Text(0.5, 1.0, 'Cross length vs Vertical length')
```



Part d:

Identify the features that have the most significant impact on fish weight and choose three of these features for further evaluation. Develop a multiple linear regression model to assess the influence of these features on fish weight and evaluate the model using

In [13]:
```
df.columns
```

Out[13]:
```
Index(['Species', 'Weight', 'Vertical', 'Diagonal', 'Cross', 'Height',
       'Width'],
      dtype='object')
```

In [14]:
```
from sklearn import linear_model
from sklearn.model_selection import train_test_split

df_encoded = pd.get_dummies(df, columns=['Species'])
# Display the resulting DataFrame
df_encoded
```

Out[14]:

	Weight	Vertical	Diagonal	Cross	Height	Width	Species_Bream	Species_Parkki	Species_Perch	Species_Pike	Species_Ro
0	242.0	23.2	25.4	30.0	11.5200	4.0200	True	False	False	False	F
1	290.0	24.0	26.3	31.2	12.4800	4.3056	True	False	False	False	F
2	340.0	23.9	26.5	31.1	12.3778	4.6961	True	False	False	False	F
3	363.0	26.3	29.0	33.5	12.7300	4.4555	True	False	False	False	F
4	430.0	26.5	29.0	34.0	12.4440	5.1340	True	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	
154	12.2	11.5	12.2	13.4	2.0904	1.3936	False	False	False	False	F
155	13.4	11.7	12.4	13.5	2.4300	1.2690	False	False	False	False	F
156	12.2	12.1	13.0	13.8	2.2770	1.2558	False	False	False	False	F
157	19.7	13.2	14.3	15.2	2.8728	2.0672	False	False	False	False	F
158	19.9	13.8	15.0	16.2	2.9322	1.8792	False	False	False	False	F

159 rows × 13 columns

In [15]:
```
correlations = df_encoded.corr()['Weight']
species_correlation = correlations.filter(like='Species').abs().mean()
```

```
correlations.drop(['Species_Bream', 'Species_Parkki', 'Species_Perch', 'Species_Pike', 'Species_Roach', 'Species_Sme

# Create a new Series with the species correlation
species_corr_series = pd.Series(species_correlation, index=['species_correlation'])

# Concatenate the two Series
all_correlations = pd.concat([correlations, species_corr_series])

# Display the correlation coefficients
print(all_correlations)
```

```
Weight                 1.000000
Vertical               0.915712
Diagonal               0.918618
Cross                  0.923044
Height                 0.724345
Width                  0.886507
species_correlation    0.218461
dtype: float64
```

As we can see, the features 'Vertica' ,'Diagonal' and 'Cross' have the highest correlation with our desired parameter Weight. So we just include them as our input features to multiple linear regression model

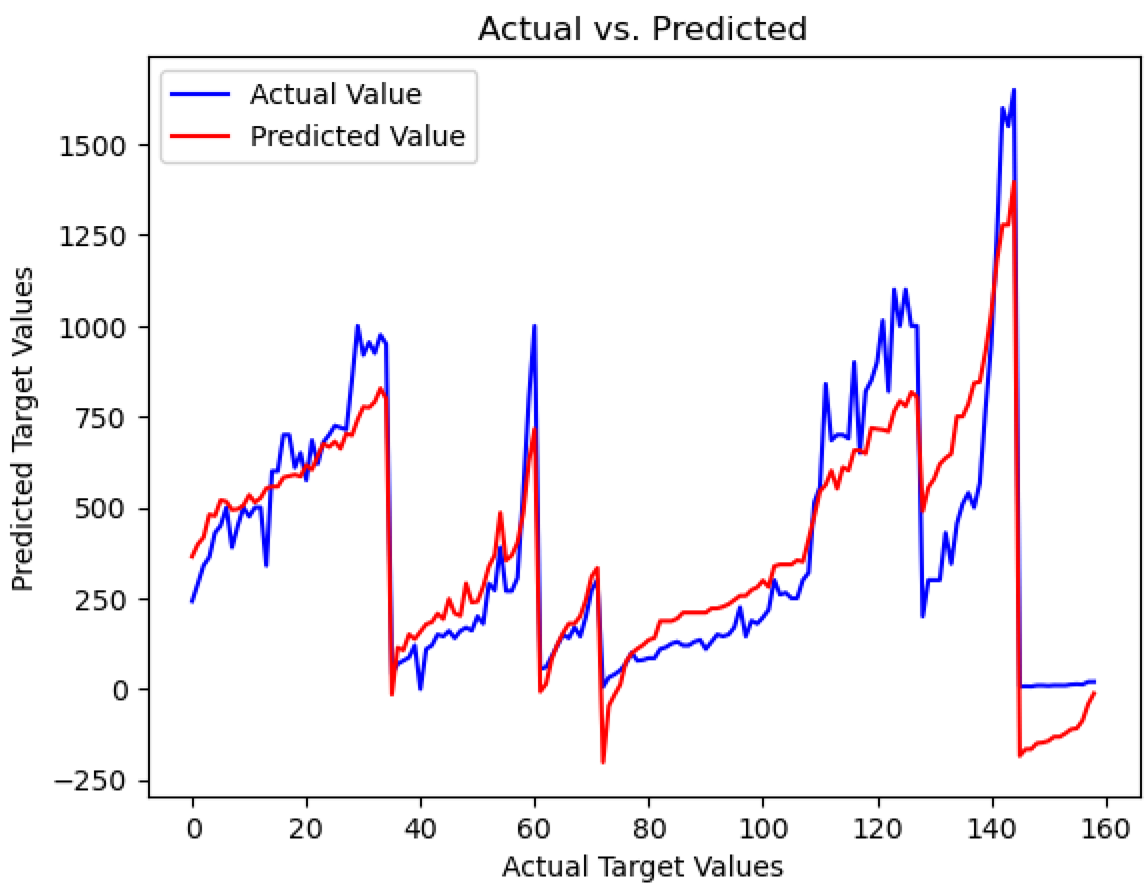
In [16]:
```
X2 = df[['Vertical', 'Diagonal', 'Cross']]
Y2 = df['Weight']
```

In [17]:
```
model = linear_model.LinearRegression()
X_train, X_test, y_train, y_test = train_test_split(X2, Y2, test_size=0.3 , random_state=101)

model.fit(X_train,y_train)
print ('Coefficients: ', model.coef_)
print ('Intercept: ',model.intercept_)
```

```
Coefficients:  [-67.2880506   70.48272872  20.0493793 ]
Intercept:  -465.6406335186988
```

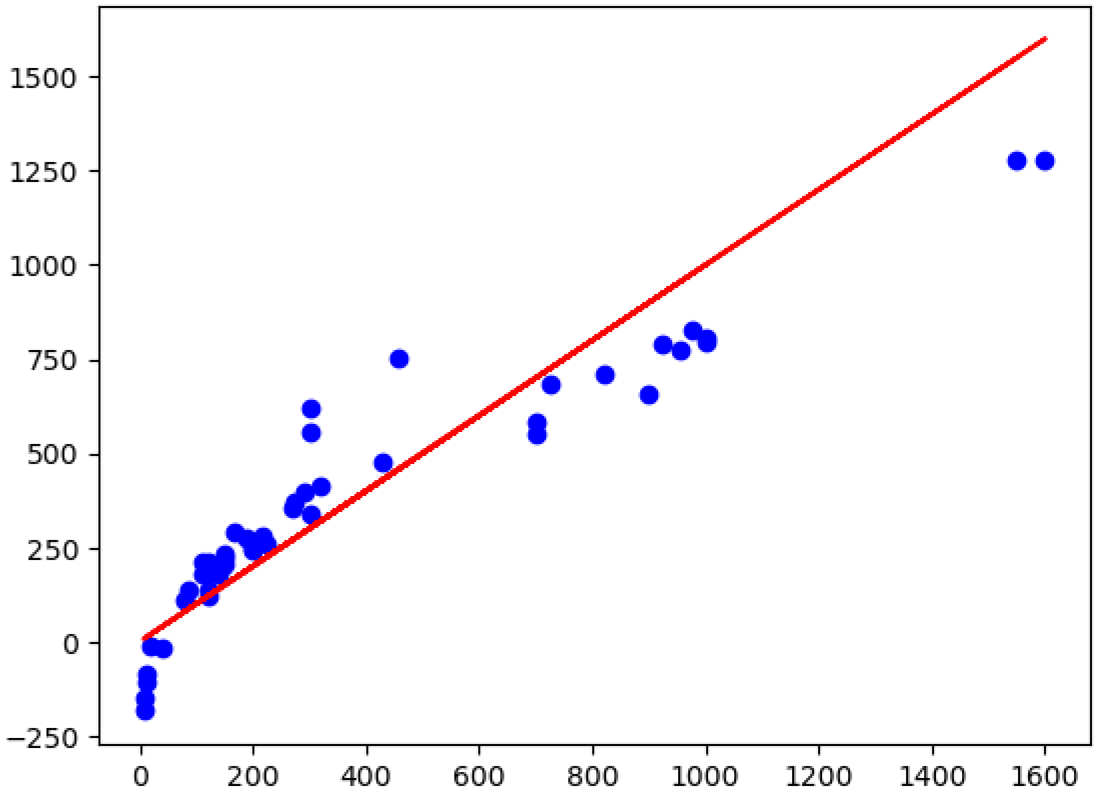
In [18]:
```
import matplotlib.pyplot as plt

# Assuming Y2 contains your actual target values and model.predict(X2) contains predicted values
plt.plot(Y2.index , Y2 ,                  color='blue'   ,  label = 'Actual Value')
plt.plot(Y2.index , model.predict(X2) , color = 'red'  , label = 'Predicted Value')
```

```
plt.xlabel('Actual Target Values')
plt.ylabel('Predicted Target Values')
plt.title('Actual vs. Predicted')
plt.legend()
plt.show()
```



In [19]:
```
plt.scatter(y_test , model.predict(X_test) , color = 'blue')
plt.plot(y_test , y_test , color = 'red')
```

Out[19]: [<matplotlib.lines.Line2D at 0x2a6dc89c2e0>]



In [20]:
```
predictions2 = model.predict(X_test)
```

In [21]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions2))
print('MSE:', metrics.mean_squared_error(y_test, predictions2))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions2)))
```

```
MAE: 111.75885382922951
MSE: 19037.996564923877
RMSE: 137.97824670912397
```

Part e:

Develop a polynomial regression model for (weight vs. width), as well as (weight vs. height).

In [22]:
```
X3 = df['Width']
Y3 = df['Weight']
```

In [23]:
```
from sklearn.preprocessing import PolynomialFeatures
```

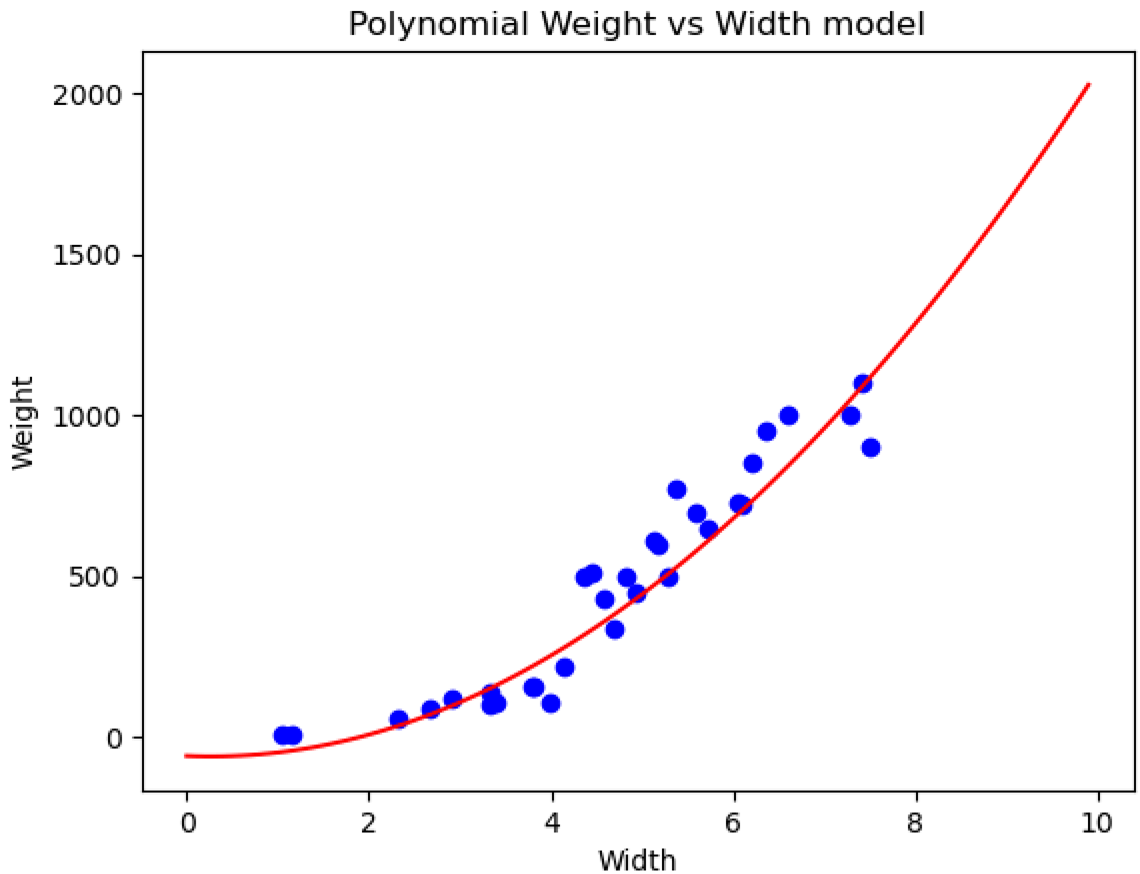
In [24]:
```
poly = PolynomialFeatures(degree = 2)
train_x, test_x , train_y , test_y = train_test_split(X3,Y3,test_size=0.2,random_state=4)
train_x_poly = poly.fit_transform(np.array(train_x).reshape(-1,1))
test_x_poly = poly.fit_transform(np.array(test_x).reshape(-1,1))
```

In [25]:
```
model = linear_model.LinearRegression()

model.fit(train_x_poly, train_y)

# The coefficients
print ('Coefficients: ', model.coef_)
print ('Intercept: ', model.intercept_)
```

```
Coefficients:  [  0.         -11.88933321  22.45259053]
Intercept:  -57.79725688019863
```

In [26]:
```
plt.scatter(test_x , test_y, color = 'blue')

XX = np.arange(0.0, 10.0, 0.1)
yy = model.intercept_+ model.coef_[1]*XX + model.coef_[2]*np.power(XX, 2)

plt.plot(XX, yy, '-r' )
plt.xlabel('Width')
plt.ylabel('Weight')
plt.title('Polynomial Weight vs Width model')
```

Out[26]:
```
Text(0.5, 1.0, 'Polynomial Weight vs Width model')
```



In [27]:
```
predictions3 = model.predict(test_x_poly)

print('MAE:', metrics.mean_absolute_error(test_y, predictions3))
print('MSE:', metrics.mean_squared_error(test_y, predictions3))
print('RMSE:', np.sqrt(metrics.mean_squared_error(test_y, predictions3)))
```

```
MAE: 81.8972986743567
MSE: 10922.507827726185
RMSE: 104.51080244513571
```

In [ ]: