In [ ]:

```
DAY15-30
Assignment Peoblems
```

In [ ]:

```
Fish Market
```

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
df = pd.read_csv(r"C:\Users\Kesam\Downloads\Kishorereddy\Fish.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	Species	Weight	Length1	Length2	Length3	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

In [5]:

```
df.columns = ['species', 'weight','vertical_length','diagnol_length','cross_length','height
df.head()
```

Out[5]:

	species	weight	vertical_length	diagnol_length	cross_length	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

In [6]:

```
df.describe()
```

Out[6]:

	weight	vertical_length	diagnol_length	cross_length	height	width
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	398.326415	26.247170	28.415723	31.227044	8.970994	4.417486
std	357.978317	9.996441	10.716328	11.610246	4.286208	1.685804
min	0.000000	7.500000	8.400000	8.800000	1.728400	1.047600
25%	120.000000	19.050000	21.000000	23.150000	5.944800	3.385650
50%	273.000000	25.200000	27.300000	29.400000	7.786000	4.248500
75%	650.000000	32.700000	35.500000	39.650000	12.365900	5.584500
max	1650.000000	59.000000	63.400000	68.000000	18.957000	8.142000

In [7]:

```
x = np.array(df['vertical_length'])
y = np.array(df['weight'])
n = len(x)
```

In [8]:

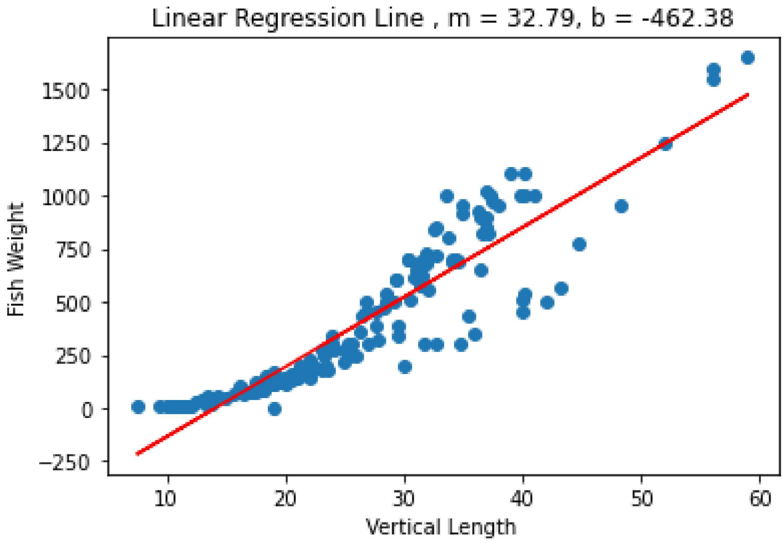
```
sum_x  = np.sum(x)
sum_y  = np.sum(y)
sum_xx = np.sum(x*x)
sum_xy = np.sum(x*y)
mean_x = np.mean(x)
mean_y = np.mean(y)

numerator   = sum_xy - ((sum_x*sum_y)/n)
denominator = sum_xx - ((sum_x*sum_x)/n)
```

In [9]:

```
m = numerator/denominator
b = mean_y - m*mean_x
```

In [10]:

```
plt.scatter(x,y)
plt.plot(x, m*x+b, color='red')
plt.title(f'Linear Regression Line , m = {m:.2f}, b = {b:.2f}')
plt.ylabel("Fish Weight")
plt.xlabel("Vertical Length")
plt.figure(figsize=(10,5))
plt.show()
```




```
<Figure size 720x360 with 0 Axes>
```

In [ ]:

```
```

In [ ]:

```
```

In [11]:

```
corr = df.corr(method='pearson')
corr
```

Out[11]:

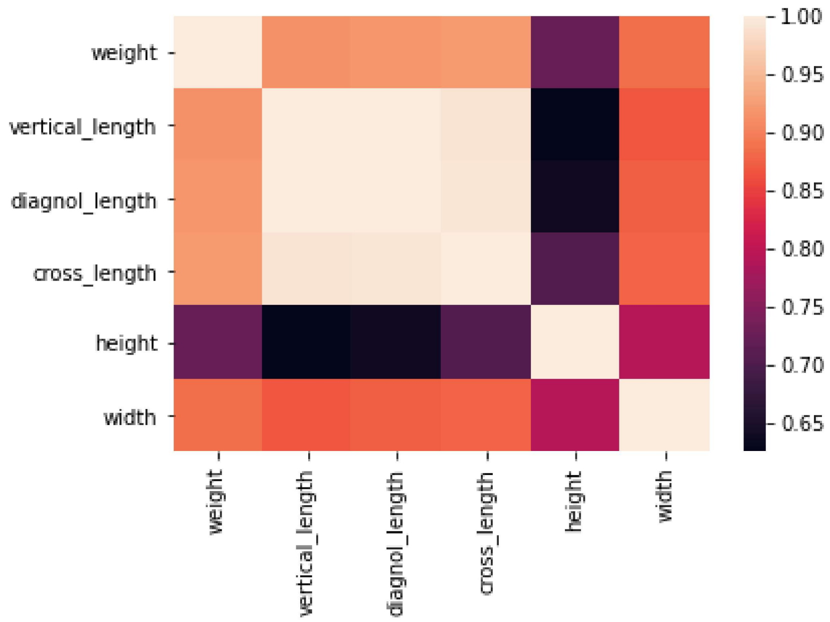
	weight	vertical_length	diagnol_length	cross_length	height	width
weight	1.000000	0.915712	0.918618	0.923044	0.724345	0.886507
vertical_length	0.915712	1.000000	0.999517	0.992031	0.625378	0.867050
diagnol_length	0.918618	0.999517	1.000000	0.994103	0.640441	0.873547
cross_length	0.923044	0.992031	0.994103	1.000000	0.703409	0.878520
height	0.724345	0.625378	0.640441	0.703409	1.000000	0.792881
width	0.886507	0.867050	0.873547	0.878520	0.792881	1.000000

In [12]:

```
# Plotting the heat map for the correlation values

sns.heatmap(corr, xticklabels = corr.columns.values, yticklabels=corr.columns.values)
```

Out[12]:

```
<AxesSubplot:>
```



In [ ]:

```
```

In [13]:

```
from sklearn.model_selection import train_test_split
x = df.drop(['weight','species'], axis=1).values
y = df['weight'].values

X_train, X_test, Y_train, Y_test =  train_test_split( x,y,test_size=0.3, random_state=10)
```

In [14]:

```
print(f'Shape of training set for X = {X_train.shape}')
print(f'Shape of testing  set for X = {X_test.shape}')
print(f'Shape of training set for Y = {Y_train.shape}')
print(f'Shape of testing  set for Y = {Y_test.shape}')
```

```
Shape of training set for X = (111, 5)
Shape of testing  set for X = (48, 5)
Shape of training set for Y = (111,)
Shape of testing  set for Y = (48,)
```

In [15]:

```
from sklearn.linear_model import LinearRegression

linear_regression = LinearRegression()
linear_regression.fit(X_train, Y_train)
Y_predict = linear_regression.predict(X_test)
```

In [ ]:

```

```

In [16]:

```
import math

from sklearn.metrics import r2_score
r2 = r2_score(Y_test ,Y_predict)
r  = math.sqrt(r2)

print(f'coefficient of determination = {r2:.3f}')
print(f'     correlation coefficient = {r:.3f}')
```

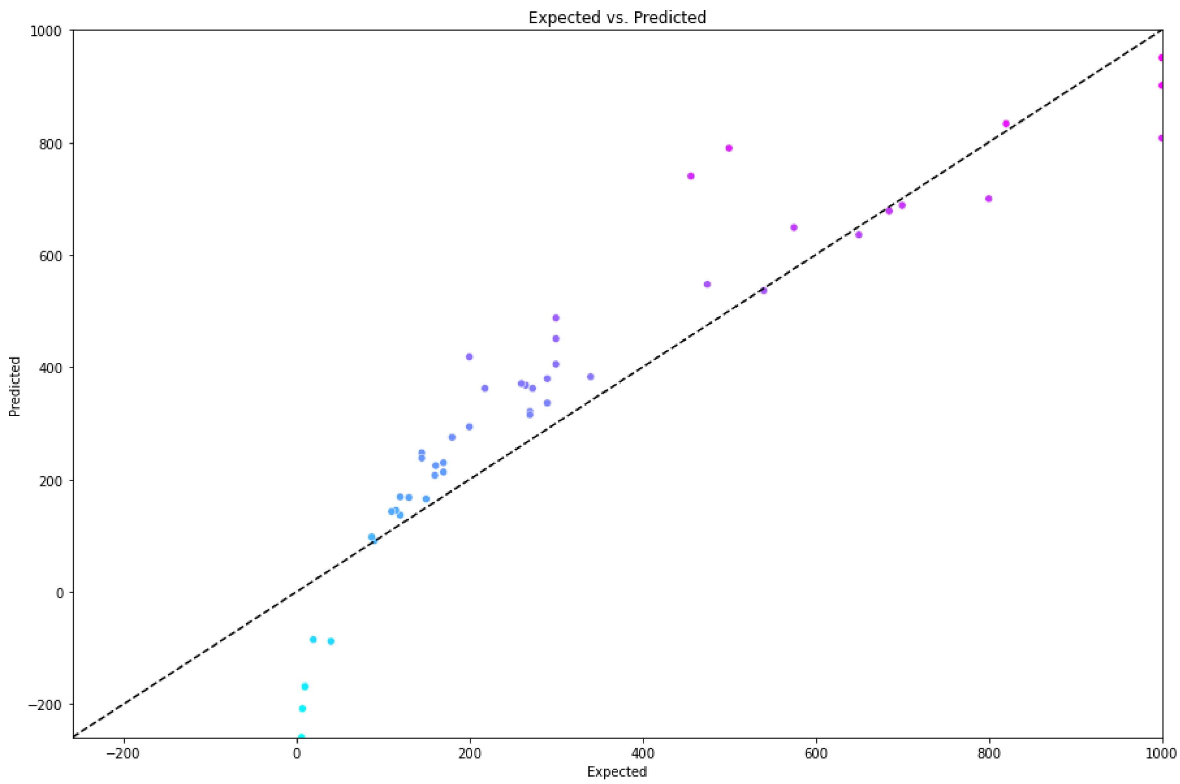
```
coefficient of determination = 0.809
     correlation coefficient = 0.899
```

In [18]:

```
from sklearn import metrics
# Using Predicted values
predicted = Y_predict
expected  = Y_test
```

In [19]:

```
df = pd.DataFrame()

df['Expected']  = pd.Series(expected)
df['Predicted'] = pd.Series(predicted)

figure = plt.figure(figsize=(15, 10))

axes = sns.scatterplot(data=df, x='Expected', y='Predicted',
                       hue='Predicted', palette='cool',
                       legend=False)

start = min(expected.min(), predicted.min())
end   = max(expected.max(), predicted.max())

axes.set_xlim(start, end)
axes.set_ylim(start, end)

plt.title('Expected vs. Predicted')
line = plt.plot([start, end], [start, end], 'k--')
```



In [ ]:

```
Vehicle dataset
```

In [26]:

```
cars_data=pd.read_csv(r'C:\Users\Kesam\Downloads\Kishorereddy\CAR DETAILS FROM CAR DEKHO.cs
```

In [27]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn import metrics
```

In [28]:

```
#checking the shape
cars_data.shape
```

Out[28]:

```
(4340, 8)
```

In [29]:

```
#getting some info of the data set
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   name           4340 non-null   object
 1   year           4340 non-null   int64 
 2   selling_price  4340 non-null   int64 
 3   km_driven      4340 non-null   int64 
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

In [30]:

```
# checking for number of missing values
cars_data.isnull().sum()
```

Out[30]:

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

In [31]:

```
#checking the distribution of categorical data
print(cars_data.fuel.value_counts())
print(cars_data.seller_type.value_counts())
print(cars_data.transmission.value_counts())
print(cars_data.owner.value_counts())
```

```
Diesel      2153
Petrol      2123
CNG           40
LPG           23
Electric       1
Name: fuel, dtype: int64
Individual          3244
Dealer               994
Trustmark Dealer     102
Name: seller_type, dtype: int64
Manual       3892
Automatic     448
Name: transmission, dtype: int64
First Owner                 2832
Second Owner                1106
Third Owner                  304
Fourth & Above Owner          81
Test Drive Car                17
Name: owner, dtype: int64
```

In [ ]:

```

```

In [32]:

```
ing fuel column
ta.replace({'fuel':{'Diesel':0,'Petrol':1,'CNG':2,'LPG':3,'Electric':4}},inplace=True)
ing transmission column
ta.replace({'transmission':{'Manual':0,'Automatic':1}},inplace=True)
ing seller_type column
ta.replace({'seller_type':{'Individual':0,'Dealer':1,'Trustmark Dealer':2}},inplace=True)
ng owner column
ta.replace({'owner':{'First Owner':0,'Second Owner':1,'Third Owner':2,'Fourth & Above Owner'
```

In [33]:

```
cars_data.head()
```

Out[33]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	1	0	0	0
1	Maruti Wagon R LXI Minor	2007	135000	50000	1	0	0	0
2	Hyundai Verna 1.6 SX	2012	600000	100000	0	0	0	0
3	Datsun RediGO T Option	2017	250000	46000	1	0	0	0
4	Honda Amaze VX i-DTEC	2014	450000	141000	0	0	0	1

In [ ]:

```

```

In [34]:

```
X=cars_data.drop(['name','selling_price'],axis=1)
Y=cars_data['selling_price']
```

In [35]:

```
X_train, X_test, Y_train, Y_test= train_test_split(X,Y,test_size=0.2,random_state=2)
```

In [ ]:

```
Model Training
```

In [ ]:

```
Linear Regression
```

In [36]:

```
# loading linear regression model
lin_reg_model=LinearRegression()
lin_reg_model.fit(X_train,Y_train)
```

Out[36]:

```
LinearRegression()
```

In [ ]:

```
model evaluation
```

In [37]:

```
#predicting
predicted_values=lin_reg_model.predict(X_train)
```

In [38]:

```
# R square error
error_score=metrics.r2_score(Y_train,predicted_values)
print("R square error:",error_score)
```

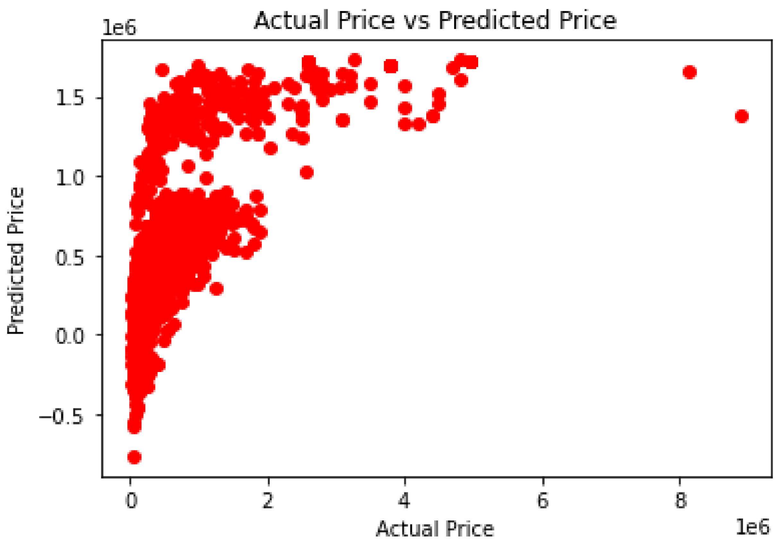
```
R square error: 0.439224106370489O3
```

In [ ]:

```
visualize actual and predicted values
```

In [39]:

```
plt.scatter(Y_train,predicted_values,c='r')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Price vs Predicted Price')
plt.show()
```



In [40]:

```
prediction_values=lin_reg_model.predict(X_test)
```

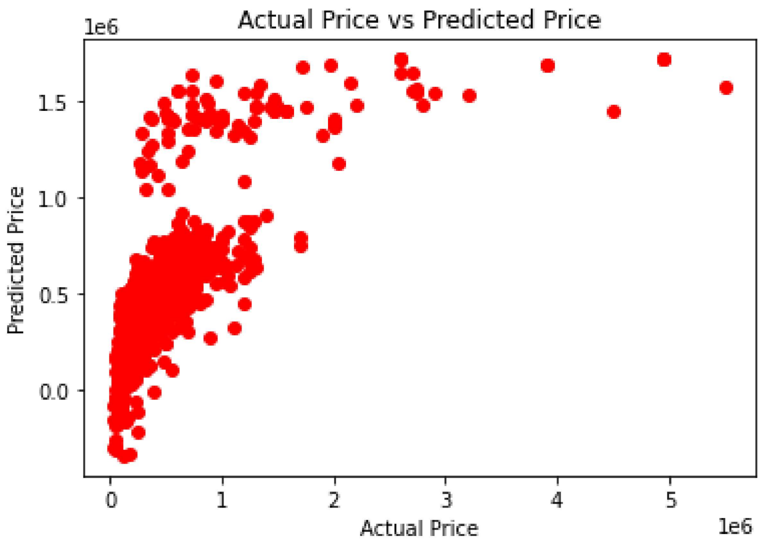
In [41]:

```
error_score1=metrics.r2_score(Y_test,prediction_values)
print("R square error:",error_score1)
```

```
R square error: 0.49988298730549363
```

In [42]:

```
plt.scatter(Y_test,prediction_values,c='r')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Price vs Predicted Price')
plt.show()
```



In [ ]:

```

```